

*DEMETRA - 3D prEcision farMing using intErnet of Things  
and unmanned aeRiAl vehicles in greenhouses*

**DELIVERABLE D3.2**  
**INTEGRATION AND TESTING RESULTS AND**  
**PERFORMANCE ANALYSIS**  
**DECEMBER 31, 2023**

**STELIOS IOANNOU, MARIOS RASPOPOULOS, ANDREY SESYUK AND  
DEMETRIS KALLASIDES**  
INTERDISCIPLINARY SCIENCE PROMOTION & INNOVATIVE RESEARCH EXPLORATION (INSPIRE)



**Co-funded by  
the European Union**



**RESEARCH  
& INNOVATION  
FOUNDATION**

*"This work was co-funded by the European Union under the programme of social cohesion "THALIA 2021-2027", through Research and Innovation Foundation (Project: CONCEPT/0722/0100)".*

## Abstract

A report that will summarize the results of the final testing carried out and compare them against the set KPIs. It will also include the final performance analysis against similar results reported in the literature as well as the applicability of the technology to be exploited in commercial products/services.

# Table of Contents

|  |    |
|--|----|
| List of Figures .....  | 4  |
| List of Tables.....  | 9  |
| 1 Proposed Greenhouse Monitoring System .....                      | 10 |
| 1.1 Concept and Objectives.....                                    | 10 |
| 1.2 Temperature & Humidity sensor - DHT-11.....                    | 12 |
| 1.2.1 DHT-11 Integration.....                                      | 12 |
| 1.2.2 DHT-11 Temperature Calibration Data.....                     | 13 |
| 1.2.3 DHT-11 Humidity Calibration Data .....                       | 15 |
| 1.3 pH Sensor - SEN0161-V2 .....                                   | 16 |
| 1.3.1 pH Sensor SEN0161-V2 Integration.....                        | 16 |
| 1.3.2 pH Sensor SEN0161-V2 Calibration .....                       | 17 |
| 1.4 Air Quality Sensor - MQ-135.....                               | 18 |
| 1.4.1 Air Quality Sensor - MQ-135 Integration and Calibration..... | 19 |
| 1.4.2 References.....  | 21 |
| 1.5 Light Intensity Sensor – Custom made Using GL5516 LDR .....    | 22 |
| 1.5.1 Light Intensity Sensor Integration.....                      | 22 |
| 1.5.2 Light Intensity Sensor Calibration .....                     | 23 |
| 1.6 Soil Moisture Sensor – Custom made.....                        | 23 |
| 1.6.1 Soil Humidity Sensor Integration .....                       | 25 |
| 1.6.2 Soil Humidity Sensor Calibration .....                       | 26 |
| 1.7 Monitoring System Output Examples.....                         | 30 |
| 2 Image Processing using YOLO.....                                 | 31 |
| 2.1 Fire Detection .....   | 31 |
| 2.2 Sensitivity Analysis.....                                      | 41 |

|       |   |    |
|-------|---|----|
| 2.3   | Insect Detection .....  | 47 |
| 3     | Multispectral Imaging.....                                    | 53 |
| 3.1   | Sensitivity Analysis.....                                     | 53 |
| 3.2   | Multispectral Image Analysis.....                             | 57 |
| 3.2.1 | DJI Terra Software .....                                      | 57 |
| 3.2.2 | Open Source Software QGIS.....                                | 60 |
| 4     | Indoor Positioning .....                                      | 67 |
| 4.1   | 3D Indoor Positioning using AI Cameras.....                   | 67 |
| 4.1.1 | Experimental Setup .....                                      | 67 |
| 4.1.2 | Distance acquisition with stereo cameras .....                | 69 |
| 4.1.3 | Stereos Depth Mapping .....                                   | 71 |
| 4.1.4 | Stereo camera calibration .....                               | 71 |
| 4.1.5 | Conversion from Camera to Local Coordinate System .....       | 72 |
| 4.1.6 | Experimental Results .....                                    | 72 |
| 4.1.7 | Analysis of Experimental Data and Accuracy Calculations.....  | 77 |
| 4.2   | 3D Positioning Indoor using mmWave Radar Sensors.....         | 78 |
| 4.2.1 | 2-DOF Setup Results .....                                     | 80 |
| 4.2.1 | 3D Multilateration Approach.....                              | 81 |
| 4.2.2 | 3D Triangulation Approach.....                                | 84 |
| 5     | Indoor Positioning by GPS Conversion and Retransmission ..... | 86 |
| 5.1   | Experimental Setup .....                                      | 86 |
| 5.1.1 | STATIC GPS SPOOFING.....                                      | 87 |
| 5.1.2 | DYNAMIC GPS SPOOFING .....                                    | 91 |
| 5.2   | Experimental Results .....                                    | 96 |



## List of Figures

|   |    |
|---|----|
| Figure 1: Concept Greenhouse Monitoring System .....                                  | 10 |
| Figure 2: Small Scale Greenhouse Prototype.....                                       | 11 |
| Figure 3: Temperature & Humidity sensor - DHT-11 .....                                | 12 |
| Figure 4: DHT-11 Integration Flowchart .....  | 12 |
| Figure 5: Ambient (Outside) Temperature & Humidity displayed on the uLCD screen ..... | 13 |
| Figure 6: Inside Temperature & Humidity displayed on the uLCD screen .....            | 13 |
| Figure 7: DHT-11 Temperature Calibration Data .....                                   | 14 |
| Figure 8: Calibration Data %Error .....   | 14 |
| Figure 9: DHT-11 Humidity Calibration Data .....                                      | 15 |
| Figure 10: Calibration Data %Error .....  | 15 |
| Figure 11: SEN0161-V2 Board Pinout .....  | 16 |
| Figure 12: pH Sensor Integration Flowchart.....                                       | 16 |
| Figure 13: Example of Display Message with pH = 4 .....                               | 17 |
| Figure 14: Example of Display Message with pH = 7 .....                               | 17 |
| Figure 15: pH Sensor SEN0161-V2 Calibration Data .....                                | 17 |
| Figure 16: pH Calibration Data %Error .....   | 18 |
| Figure 17: MQ-135 Pin mapping.....  | 18 |
| Figure 18: MQ-135 Air quality sensor Flowchart.....                                   | 19 |
| Figure 19: ppm value displayed on the uLCD screen.....                                | 20 |
| Figure 20: Sensitivity characteristics of the MQ-135 for several gases [2] .....      | 20 |
| Figure 21: PPM & Rs/Ro correlation graph for CO2.....                                 | 21 |
| Figure 22: LDR in voltage divider configuration.....                                  | 22 |
| Figure 23: LDR Sensor Flowchart .....   | 22 |
| Figure 24: Example of Light Intensity Sensor Output .....                             | 23 |
| Figure 25: Soil Moisture Circuit .....  | 24 |
| Figure 26: Soil Humidity Experimental Setup .....                                     | 24 |
| Figure 27: Soil Humidity Sensor Integration Flowchart .....                           | 25 |
| Figure 28: Soil moisture displayed on the uLCD screen.....                            | 25 |
| Figure 29: Error range of each resistor .....   | 27 |
| Figure 30: Soil Resistance Experimental Data.....                                     | 28 |
| Figure 31: Results for Probes 1-2 and 1-3 .....                                       | 28 |
| Figure 32x: Results for Probes 1-4 and 3-4.....                                       | 29 |

|  |    |
|--|----|
| Figure 33: Data Analysis using 2nd Order Polynomial .....              | 29 |
| Figure 34: Data Analysis using 3rd Order Polynomial .....              | 30 |
| Figure 35: Monitoring System Output Examples .....                     | 30 |
| Figure 36: YOLO Performance Benchmark .....                            | 32 |
| Figure 37: Comparison between RBI4B and Jetson nano .....              | 33 |
| Figure 38L Proposed optimization Framework.....                        | 33 |
| Figure 39: Network Architecture .....                                  | 34 |
| Figure 40: Network Pruning.....  | 34 |
| Figure 41: Sparse training and pruning preparation.....                | 35 |
| Figure 42: Pruning process and Fine-tuning of the pruned model .....   | 35 |
| Figure 43: Hardware Acceleration Results.....                          | 36 |
| Figure 44: Fire Detection Dataset.....                                 | 37 |
| Figure 45: Comparison of models 'performances on Prediction .....      | 37 |
| Figure 46: Analysis of the model pruning results .....                 | 38 |
| Figure 47: Network Structure Comparison.....                           | 38 |
| Figure 48: Overclocking Performance Comparison .....                   | 39 |
| Figure 49: Detection results from the Comparison Experiment .....      | 39 |
| Figure 50: Detection Results on Raspberry Pi Platform .....            | 40 |
| Figure 51: Video input Results for testing the model performance ..... | 40 |
| Figure 52: FireNet Model Summary Training Results .....                | 42 |
| Figure 53: YOLOV5 Model Summary Training Results .....                 | 43 |
| Figure 54: 1080p HD quality at over 7FPS on the Raspberry Pi 4B.....   | 44 |
| Figure 55: The experiments result in the dark environments. ....       | 44 |
| Figure 56: The experiments result in the bright environments. ....     | 45 |
| Figure 57: Dataset for Insect Detection Yolo Model Training.....       | 47 |
| Figure 58: Metrics over epochs .....                                   | 48 |
| Figure 59: Precision Confidence Curve .....                            | 48 |
| Figure 60: Precision - Recall Curve .....                              | 49 |
| Figure 61: Insect Detection Results .....                              | 50 |
| Figure 62: Insect Detection Results 2 .....                            | 51 |
| Figure 63: Insect detection Results 3.....                             | 52 |
| Figure 64: Flight test Area Verification for any Restrictions .....    | 53 |
| Figure 65: Licence Certificate of UAV Test Pilot .....                 | 54 |

|   |    |
|---|----|
| Figure 66: Flight Plan .....                              | 54 |
| Figure 67: UAV Take-Off .....                             | 55 |
| Figure 68: UAV Landing .....                              | 55 |
| Figure 69: Weather Conditions the Day of Test Flight..... | 55 |
| Figure 70: Presence of Wind – Swaying Trees.....          | 56 |
| Figure 71: Hourly Weather Forecast .....                  | 56 |
| Figure 72: Terra Mapping Output – RGB.....                | 57 |
| Figure 73: Terra Mapping Output – NDVI.....               | 58 |
| Figure 74: Terra Mapping Output – GNDVI .....             | 58 |
| Figure 75: Terra Mapping Output – LCI .....               | 59 |
| Figure 76: Terra Mapping Output – NDRE.....               | 59 |
| Figure 77: Terra Mapping Output – OSAVI .....             | 60 |
| Figure 78 Visual Image 0119 .....                         | 60 |
| Figure 79 NDVI for 0119 .....                             | 61 |
| Figure 80 NDVI histogram for 0119 .....                   | 61 |
| Figure 81 NDRE for 0119 .....                             | 61 |
| Figure 82 NDRE Histogram for 0119.....                    | 61 |
| Figure 83 OSAVI for 0119 .....                            | 62 |
| Figure 84 OSAVI Histogram for 0119.....                   | 62 |
| Figure 85 NDWI for 0119.....                              | 62 |
| Figure 86 NDWI histogram for 0119.....                    | 62 |
| Figure 87 GNDVI for 0119 .....                            | 63 |
| Figure 88 GNDVI histogram for 0119 .....                  | 63 |
| Figure 89 NDVI for 0136.....                              | 63 |
| Figure 90 NDVI histogram for 0136 .....                   | 63 |
| Figure 91 NDRE for 0136 .....                             | 64 |
| Figure 92 NDRE for 0136 .....                             | 64 |
| Figure 93 OSAVI for 0136 .....                            | 64 |
| Figure 94 OSAVI histogram for 0136 .....                  | 64 |
| Figure 95 NDWI for 0136.....                              | 65 |
| Figure 96 Histogram for 0136.....                         | 65 |
| Figure 97 GNDVI for 0136 .....                            | 65 |
| Figure 98 Histogram for 0136.....                         | 65 |

|   |    |
|---|----|
| Figure 99 LCI for 0136 .....  | 66 |
| Figure 100 LCI histogram for 0136 .....   | 66 |
| Figure 101: Experimental Setup .....  | 67 |
| Figure 102: Cameras and Ground Truth Markers in 3D Space .....  | 68 |
| Figure 103: Because the depth map contains the Z distance, objects parallel to the camera are measured exactly as standard. For objects that are not parallel, the Euclidean distance can be calculated using the Euclidean distance. (Luxonis, 2022) ..... | 69 |
| Figure 104: The disparity refers to the distance between the two corresponding points in the left and right images of a stereo pair. (Luxonis, 2022) .....  | 70 |
| Figure 105: Obtaining the distance, angle offset and other information using the proposed system. ....  | 70 |
| Figure 106: Outputting the detected information to the terminal.....  | 71 |
| Figure 107: Camera and Local Coordinate Systems .....   | 72 |
| Figure 108: Example of Experimental Procedure .....   | 72 |
| Figure 109: UAV Detected by Birdeye View Camera and Location is calculated .....  | 73 |
| Figure 110: Detection of 2 UGVs Simultaneously.....   | 73 |
| Figure 111: Detection of 3 UGVs Simultaneously.....   | 74 |
| Figure 112: Profile Camera Detecting a UGV.....   | 74 |
| Figure 113: UAV Placed on a box to Verify Altitude Calculations .....   | 75 |
| Figure 114: UGV Detection at a distance of 7.5m .....   | 75 |
| Figure 115: Profile Camera Detecting 2 UGV Simultaneously .....   | 76 |
| Figure 116: Both Cameras Simultaneously Detect the Basketball.....  | 76 |
| Figure 117: Basketball is hidden from the view of Birdeye Camera .....  | 76 |
| Figure 118: Localization Error.....   | 77 |
| Figure 119: : mmWave 3D Positioning Experimental Setup using 2-DOF mmWave Sensors (Y: Yaw, P: Pitch, R: Roll)...  | 81 |
| Figure 120 - 3D Triangulation - Sensor Arrangement.....   | 84 |
| Figure 121: Ardupilot with GPS module connected .....   | 86 |
| Figure 122: 2 HackRF ONE.....   | 86 |
| Figure 123: Latest ephemeris broadcast file .....   | 87 |
| Figure 124: GPS Signal before transmission .....  | 88 |
| Figure 125: Command to generate the bin file .....  | 89 |
| Figure 126: folder organization .....   | 89 |
| Figure 127: Command for bin file transmission.....  | 90 |
| Figure 128: Mission planner after GPS transmission .....  | 90 |

|   |     |
|---|-----|
| Figure 129: Draw a custom route from google maps provider in SatGen ..... | 91  |
| Figure 130: Parameters for dynamic scenario.....                          | 92  |
| Figure 131: Dynamic KML file .....  | 93  |
| Figure 132: Dynamic bin file generation.....                              | 93  |
| Figure 133: add gpssim.bin to same directory with hackrf_transfer .....   | 94  |
| Figure 134: HackRF dynamic transmission prompt .....                      | 95  |
| Figure 135: Mission planner during dynamic GPS transmission .....         | 95  |
| Figure 136: GPS Coordinates UCLan Cyprus from Google Earth.....           | 96  |
| Figure 137: Mission Planner Before GPS Transmission .....                 | 96  |
| Figure 138: Mission Planner After GPS Transmission .....                  | 97  |
| Figure 139: Spectrum before GPS Transmission .....                        | 97  |
| Figure 140: Spectrum After GPS Transmission .....                         | 98  |
| Figure 141: Latitude GPS Received by Autopilot .....                      | 98  |
| Figure 142: Longitude GPS Received by Autopilot .....                     | 98  |
| Figure 143: Longitude versus Latitude .....                               | 99  |
| Figure 144: GPS Signal before and After Kalman Filtering.....             | 99  |
| Figure 145: Error after Kalman Filtering .....                            | 100 |
| Figure 146: Closer Look at the Error after Kalman Filtering .....         | 100 |
| Figure 147: Dynamic GPS Retransmission .....                              | 101 |

## List of Tables

|   |    |
|---|----|
| Table 1: DHT-11 Temperature Calibration Data .....                | 14 |
| Table 2: DHT-11 Humidity Calibration Data.....                    | 15 |
| Table 3: Voltage divider resistor error range.....                | 26 |
| Table 4: Resistivity of the soil in different water content ..... | 27 |
| Table 5: Location of Cameras and Ground Truth Markers .....       | 68 |
| Table 6: Localization Error .....                                 | 77 |
| Table 7 - 4 Anchor Configuration - Equal Height .....             | 82 |
| Table 8 - 4 Anchor Configuration - Different Height.....          | 82 |
| Table 9 - 6 Anchor Configuration - Different Height.....          | 83 |
| Table 10 - 3D Triangulation Positioning.....                      | 85 |

# 1 Proposed Greenhouse Monitoring System

## 1.1 Concept and Objectives

The proposed Greenhouse monitoring system was the outcome of an extensive literature review as presented in Deliverable D3.1 Technology Concept and Technical Requirements, Scenarios and KPIs. The proposed monitoring system would have the capability for measurements including Air temperature, Air humidity, Sunlight, Carbon dioxide, Soil moisture and Mineral soil composition. The concept for the monitoring system is best visualised using the block diagram on Figure 1.

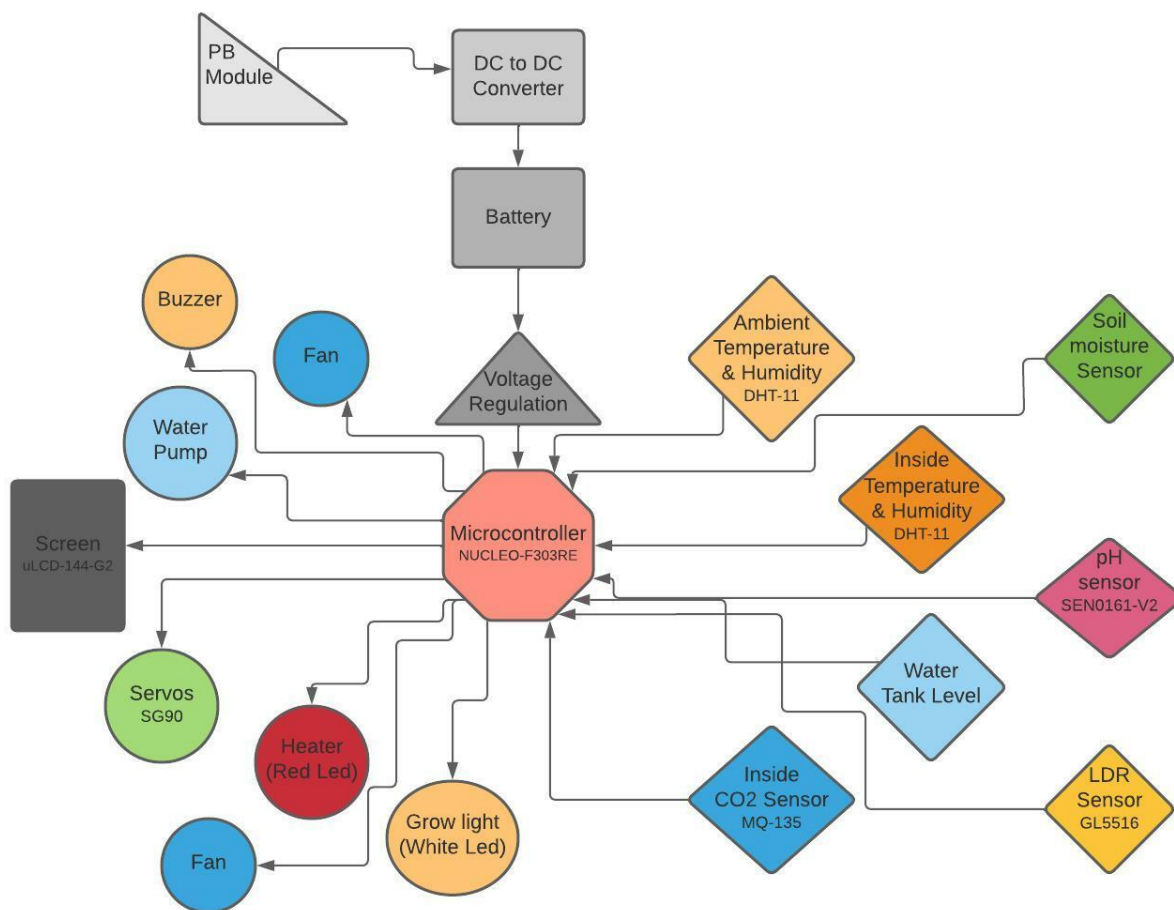


Figure 1: Concept Greenhouse Monitoring System

. The system is composed of the following main subsystems and sensors:

- Controller - STM32 Nucleo-F303RE microcontroller
- Temperature & Humidity - DHT-11
- pH Sensor - SEN0161-V2

- Air Quality Sensor - MQ-135
- Light Intensity Sensor – Custom made Using GL5516 LDR
- Soil Moisture Sensor – Custom made

Worth noting that two sensors are custom made which enables further research whose outcome are the two individual customised sensors.

In addition to the two custom made sensors, it is important to note that the block diagram includes possible future improvements that the monitoring system could offer. The future improvements are as follows:

- Measurements of temperature, humidity and light intensity should be taken from inside and outside the greenhouse. This method significantly contributes on the reduction of operating costs. For example, allow light to enter instead of turning ON artificial light sources. It also works equally well for heating and cooling by exchanging air between inside and outside accordingly.
- Actuators for opening and closing windows.
- Heaters for rising the internal temperature.
- Fans and mist sprayers for cooling.
- Buzzers to act as a warning system in the event of a fault.
- Water reservoir level monitoring to enable the spaying process.
- Capability to turn ON/OFF the water pump for spraying.
- PV Module so that the monitoring system uses rechargeable batteries.

The integration of the proposed future improvements could convert the smart greenhouse to an autonomous smart greenhouse which could significantly reduce operating costs while increasing productivity and profits.



*Figure 2: Small Scale Greenhouse Prototype*

For the purpose of sensor integration and calibration a small scale greenhouse prototype was constructed as shown on Figure 2.



## 1.2 Temperature & Humidity sensor - DHT-11

The smart greenhouse measures the internal and external temperature and humidity and displays them to the user. In order to do that the two DHT-11 temperature and humidity sensors are connected to D15 and D14 digital inputs of the microcontroller respectively.

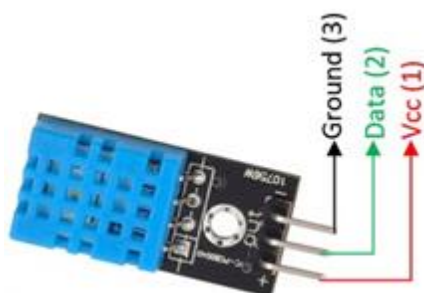


Figure 3: Temperature & Humidity sensor - DHT-11

### 1.2.1 DHT-11 Integration

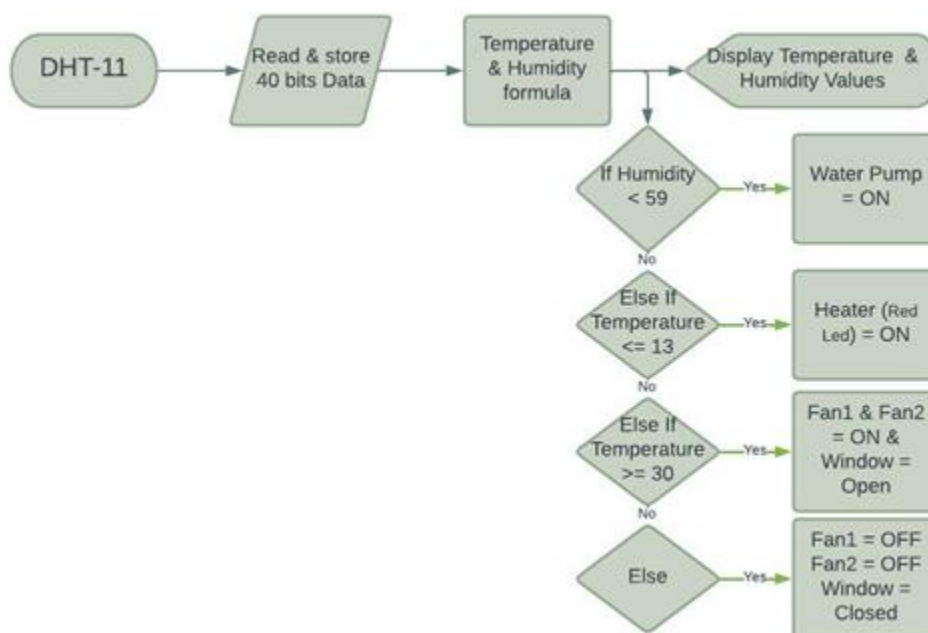


Figure 4: DHT-11 Integration Flowchart

The integration of the DHT-11 temperature and humidity sensor is best visualised using the block diagram / flowchart of Figure 4. When the data is extracted from the sensors (Fig. 4), 40 bits of data are assigned and stored to the code (Appendix 2: DHT-11 Code). Subsequently, the data enters the temperature and humidity formula (line 97-98) in order to calculate the actual temperature and humidity of internal (inside) and external (ambient) environment. Then the results are displayed on the uLCD screen (Fig. 5 and 6).



Figure 5: Ambient (Outside) Temperature & Humidity displayed on the uLCD screen



Figure 6: Inside Temperature & Humidity displayed on the uLCD screen

As figure 4 shows after the displaying of the temperature and humidity some “If” statements were situated. At the first statement, if humidity is less than 59 percent the microcontroller will activate the water pump which is installed in PC9 “pwm” (Pulse Width Modulation) pin.

The second statement clarifies that if the temperature is less or equal to 13°C the microcontroller will activate the heater, which is represented with a red led lamb. The third “If” statement makes clear that if temperature is more or equal to 30°C the microcontroller will activate “Fan1” and “Fan2” which are installed in the greenhouse and decrease the temperature. The two fans are installed in D5 and PC9 “pwm” pins respectively. Additionally the servo motor will be activated, automatically opening the window at an angle of 60 degrees. The servo is installed in D9 “pwm” pin.

The last statement is used to make sure that if none of the above statement is true, water pump, “Fan1”, “Fan2” and the servo motor are not activated and in their initial positions.

### 1.2.2 DHT-11 Temperature Calibration Data

The DHT-11 temperature and humidity sensor has been calibrated using a digital thermometer as a reference for the temperature to make sure that it functions properly and to compare the error range. Ten values were taken for each device (digital thermometer and DHT-11 sensor) and the calibration data is tabulated on Table 1 and plotted on Figures 7 and 8.

Comparing the data from Figure 7, the temperature measurements of sensor DHT-11 are very smooth and have minor deviations compared to the digital thermometer that was used for reference and comparison purposes.

Table 1: DHT-11 Temperature Calibration Data

| Measured Data | DHT11 (°C) | Digital Thermometer (°C) | Difference (%) |
|---------------|------------|--------------------------|----------------|
| 1             | 20.8       | 20                       | -4.00          |
| 2             | 20.9       | 20                       | -4.50          |
| 3             | 21         | 20                       | -5.00          |
| 4             | 21.2       | 21                       | -0.95          |
| 5             | 25         | 27                       | 7.41           |
| 6             | 26         | 28                       | 7.14           |
| 7             | 25.5       | 26                       | 1.92           |
| 8             | 23.4       | 23                       | -1.74          |
| 9             | 23         | 23                       | 0.00           |
| 10            | 21         | 20                       | -5.00          |

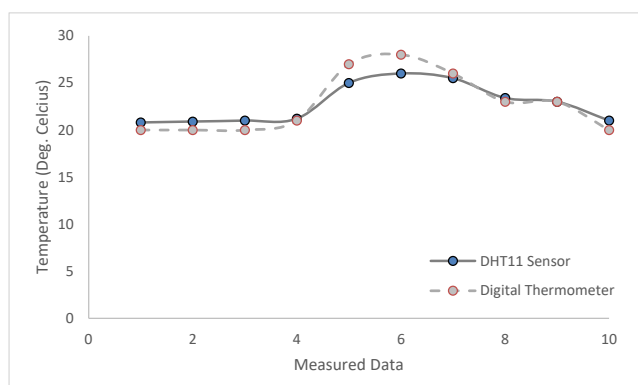


Figure 7: DHT-11 Temperature Calibration Data

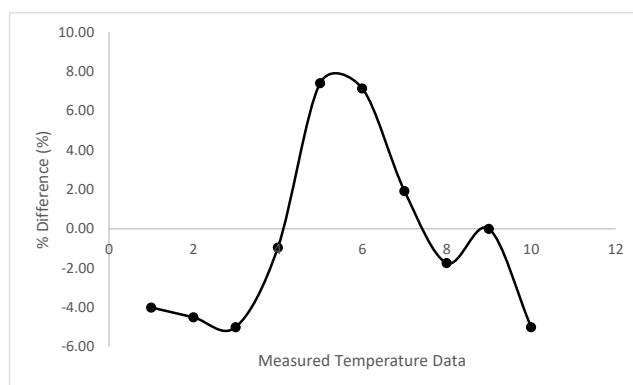


Figure 8: Calibration Data %Error

As shown on Table 1 and Figure 8, the temperature calibration data for sensor DHT-11 shows the highest percentage error being 7.4% which equates to an accuracy of +2°C. The temperature accuracy of +2°C is within the range of indicated KPIs.

### 1.2.3 DHT-11 Humidity Calibration Data

The same process was repeated for the calibration of the humidity that DHT-11 sensor. Ten samples were taken in ten different times from the DHT-11 temperature and humidity sensor. The results are tabulated on Table 3.

Table 2: DHT-11 Humidity Calibration Data

| Measured Data | Humidity DHT11 (%) | Digital RH (%) | Difference (%) |
|---------------|--------------------|----------------|----------------|
| 1             | 66                 | 69             | 4.35           |
| 2             | 45                 | 47             | 4.26           |
| 3             | 84                 | 87             | 3.45           |
| 4             | 62                 | 67             | 7.46           |
| 5             | 43                 | 45             | 4.44           |
| 6             | 85                 | 88             | 3.41           |
| 7             | 68                 | 70             | 2.86           |
| 8             | 49                 | 52             | 5.77           |
| 9             | 84                 | 89             | 5.62           |
| 10            | 78                 | 82             | 4.88           |

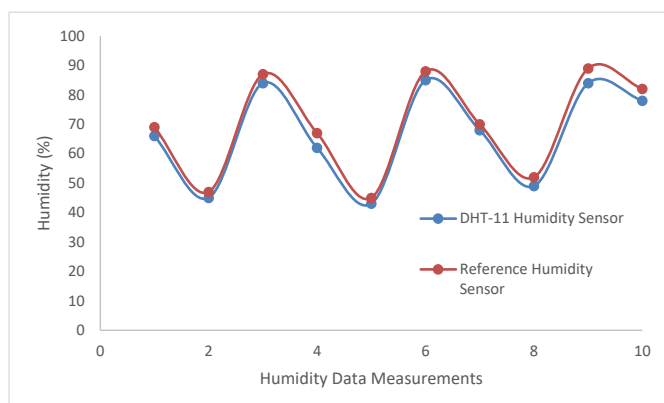


Figure 9: DHT-11 Humidity Calibration Data

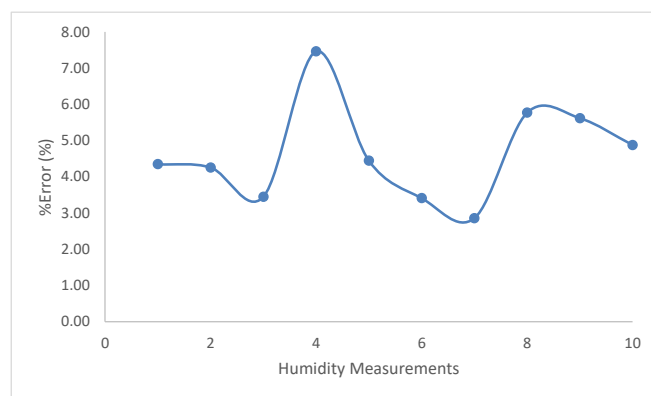


Figure 10: Calibration Data %Error

Comparing the data from Figure 9, the humidity measurements of sensor DHT-11 are very smooth and have minor deviations compared to the digital humidity sensor that was used for reference and comparison purposes.

Furthermore, as shown on Table 2 and Figure 10, the humidity calibration data for sensor DHT-11 shows the highest percentage error being 7.46% which equates to an accuracy of +5%. **The humidity accuracy of +5% is within the range of indicated KPIs.**

### 1.3 pH Sensor - SEN0161-V2

The output signal of the sensor is connected to the A2 analog input of the NUCLEO board.

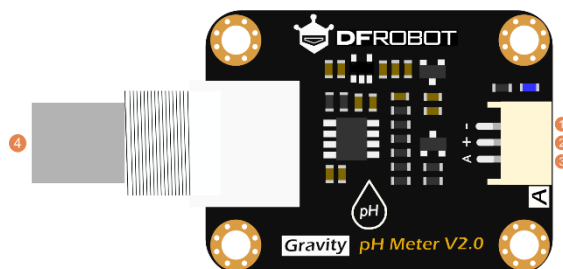


Figure 11: SEN0161-V2 Board Pinout

#### 1.3.1 pH Sensor SEN0161-V2 Integration

The integration of the pH Sensor SEN0161-V2 sensor is best visualised using the block diagram / flowchart of Figure 12.

The code stores an average result of ten values (Signal Conditioning) and then pH value is calculated (Data Analysis).

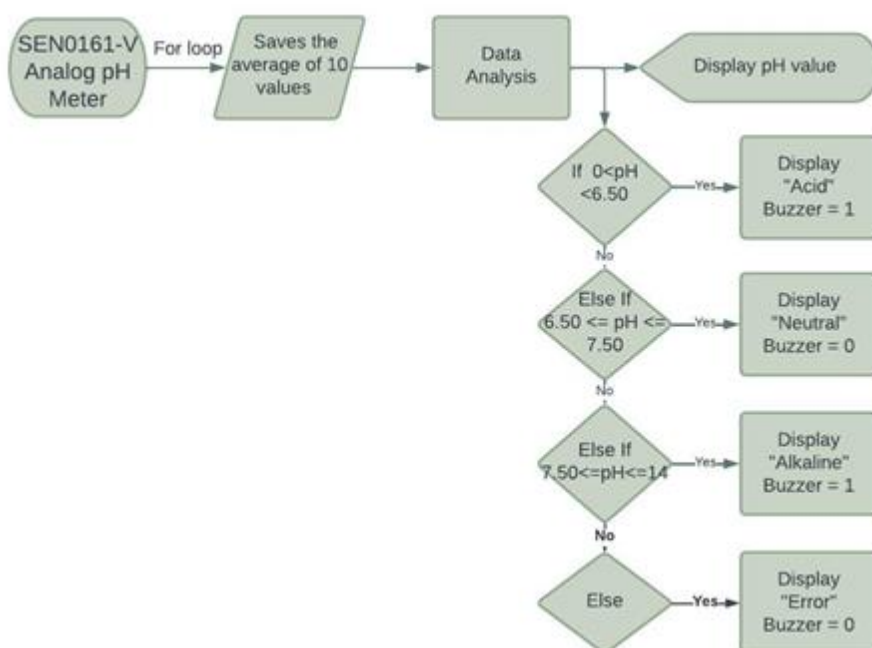


Figure 12: pH Sensor Integration Flowchart

If the pH value ranges from 0 to 6.50, then an “Acid” sign appears in the screen and the buzzer gets activated to inform the user. If the pH value ranges from 6.50 up to 7.50, the buzzer is disabled and the word “Neutral” will appear on the uLCD screen. Another possibility is for the pH value to range from 7.50 up to 14, which will result to the activation of the buzzer and the word “Alkaline” appearing on the screen. If none of the above statements are true, then an “Error”

sign will appear in the screen and the buzzer is disabled. Examples of the output messages displayed are shown on figures 13 and 14.



Figure 13: Example of Display Message with pH = 4



Figure 14: Example of Display Message with pH = 7

### 1.3.2 pH Sensor SEN0161-V2 Calibration

In order to calibrate the SEN0161-V2 analog pH sensor, two different liquids with different pH level (4 and 7) that were included in the box of the sensor were used. After connecting the pH sensor to the microcontroller, two analogue readings from the two liquids were collected. From liquid 1 that has a value of 4.0 pH, a value of 0.620V was measured using a digital multi-meter whereas for liquid 2 that has a value of 7.0 pH, a value of 0.460V was measured. At the same time the microcontroller was reporting input voltages of 0.605V and 0.446V respectively. The data was then was plotted as shown on Figure 15.

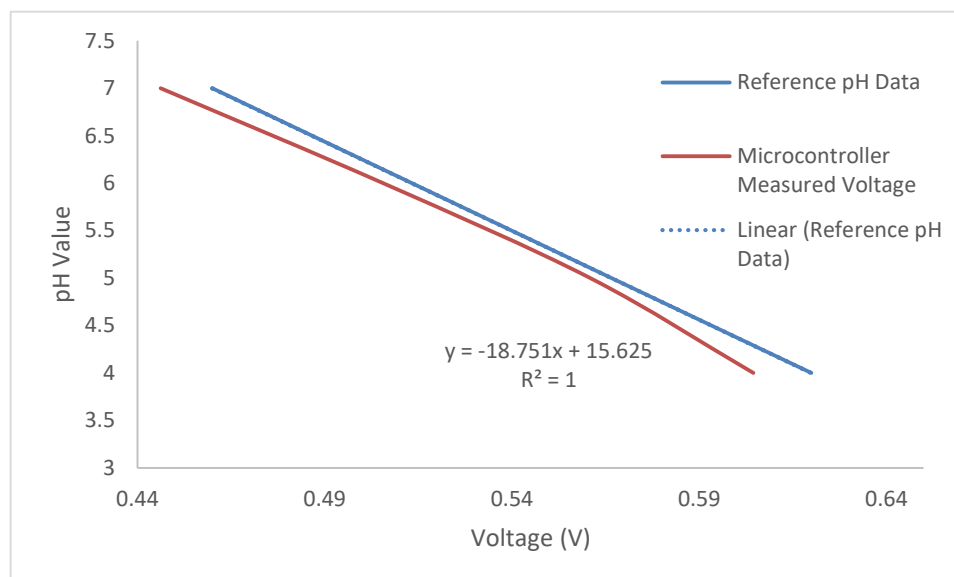


Figure 15: pH Sensor SEN0161-V2 Calibration Data

From the results presented on Figure 15, then the pH model equation was derived as shown on equation 1.

$$pH = -18.751x + 15.625 \quad (1)$$

Where  $x$  is the voltage from the sensor that is read by the microcontroller. In addition, as presented by the graphs trendline, the linear equation fits the data 100%.

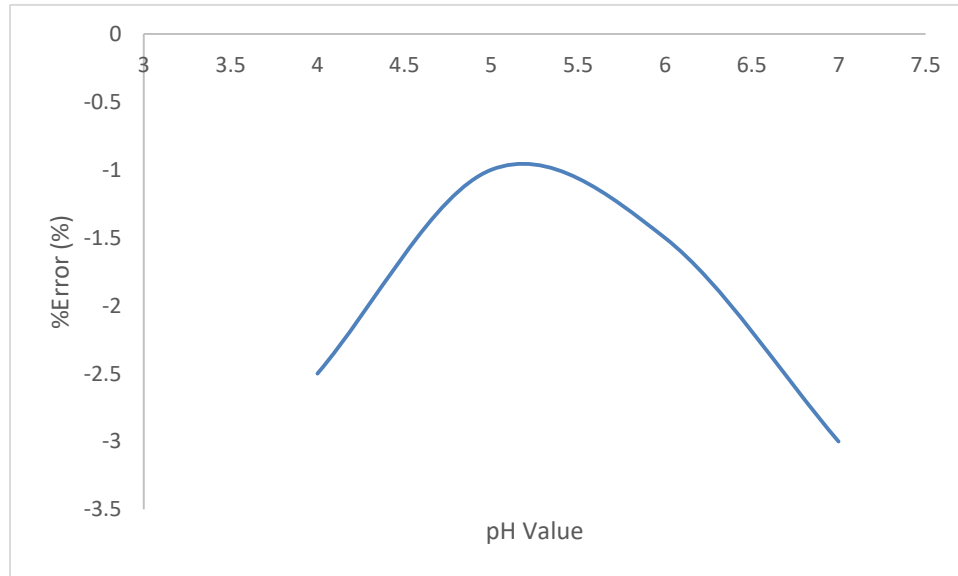


Figure 16: pH Calibration Data %Error

However, as seen from Figure 16, even though the mathematical modeling fits the data 100%, yet due to ADC (Analogue to Digital Conversion) and DAC (Digital to Analogue Conversion) conversion in the microcontroller, a maximum error of 3% is introduced. **The pH accuracy of +5% is within the range of indicated KPIs.**

#### 1.4 Air Quality Sensor - MQ-135

MQ-135 is an air quality sensor that is suitable for detecting gases like Ammonia ( $\text{NH}_3$ ), Sulfur (S), Benzene ( $\text{C}_6\text{H}_6$ ),  $\text{CO}_2$  and other harmful gases. When the level of these gases surpasses a threshold limit in the air, the analogue output pin, outputs an analogue voltage which can be used to extract the level of these gases in the atmosphere. The MQ-135 operates from 2.5V up to 5.0V and it consumes about 150mA. The sensor provides both digital and analog output.

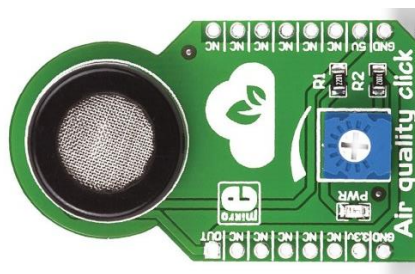


Figure 17: MQ-135 Pin mapping

Pins one and three of the MQ-135 air quality sensor, are connected to the ground of the microcontroller, whereas pin 2 is connected to the voltage supply of 5V, and finally pin four is connected to an analogue pin of the microcontroller where it extracts the raw data of the sensor. In order for the MQ-135 sensor to extract the correct values, it needs to power up for a pre-heat duration. This pre-heat time is normally between 30 seconds to a couple of minutes.

#### 1.4.1 Air Quality Sensor - MQ-135 Integration and Calibration

The Analogue output pin of the sensor can be used to measure the PPM (Parts per Million) value of a specific gas. For the purposes of this project the gas chosen was CO<sub>2</sub>. The analogue output signal of the sensor is connected to the A0 analogue pin of the microcontroller. For filtering purposes, the code stores an average result of ten measurements and displays the CO<sub>2</sub> value.

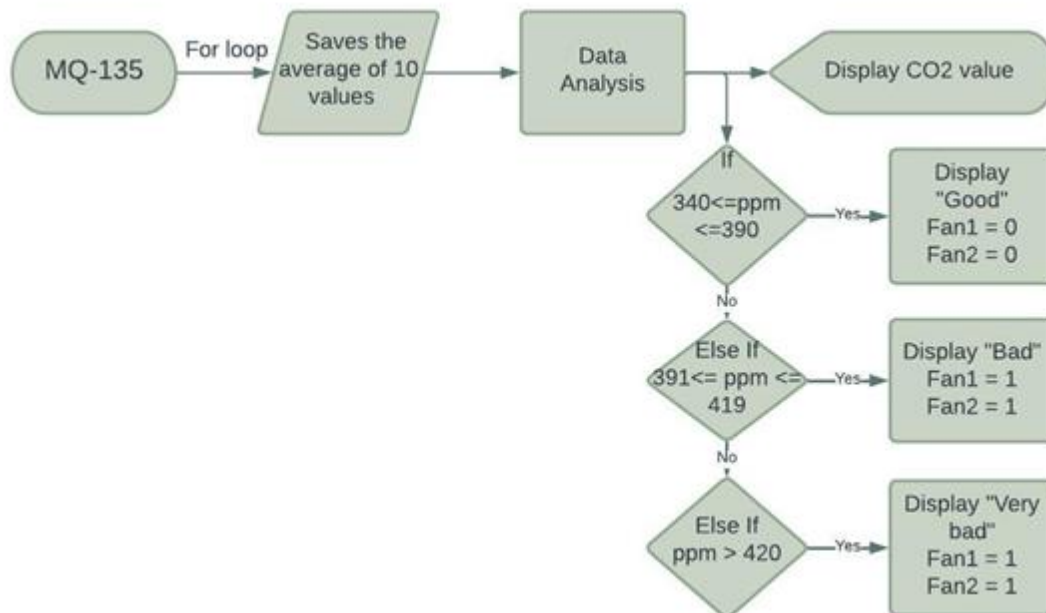


Figure 18: MQ-135 Air quality sensor Flowchart

If the CO<sub>2</sub> value ranges from 340 ppm to 390 ppm, then a “Good” sign appears on the screen and the two fans (Fan1, Fan2) are disabled. If the CO<sub>2</sub> value ranges from 391 ppm to 419 ppm, the two fans are activated and the word “Bad” will appear on the screen. Another possibility is for the CO<sub>2</sub> value to exceed the 420 ppm, which will result to the activation of the two fans and the words “Very bad” appearing on the uLCD screen (figure 19).





Figure 19: ppm value displayed on the uLCD screen

Using the microcontroller, a measurement of the analog voltage value can be extracted and execute some calculations to acquire the resistance ratio of the sensor ( $R_s/R_o$ ).  $R_s$  is the sensor resistance when a gas is present and  $R_o$  is sensor resistance at clean air. Another coefficient is the  $R_L$  that is the pull-down resistor of the MQ-135 air quality sensor. Once this ratio of  $R_s/R_o$  is retrieved, it can be used to calculate the PPM value of the required gas using the graph (Figure 20) below which represents the concentration of a gas in part per million (ppm) of the MQ-135 Sensor for several gases [1].

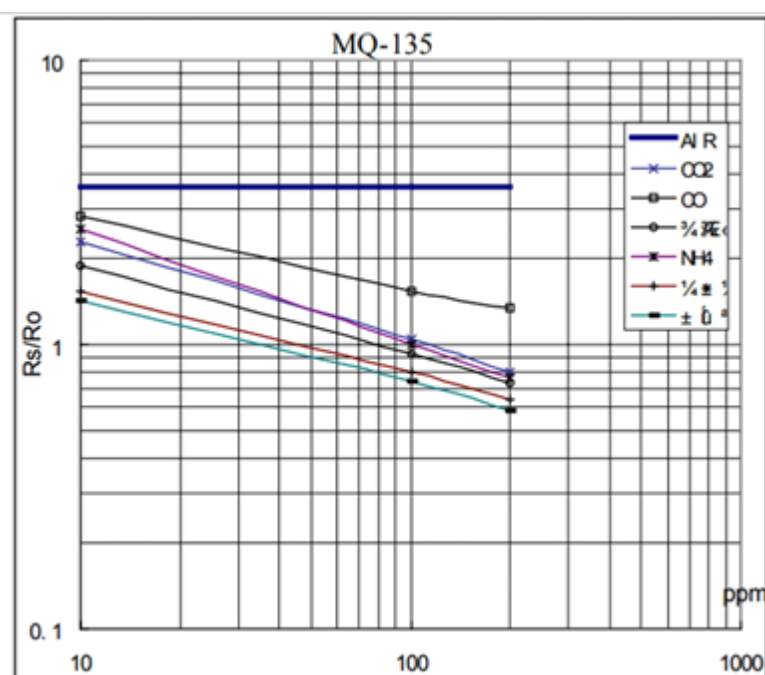


Figure 20: Sensitivity characteristics of the MQ-135 for several gases [2]

In order to calibrate the MQ-135 air quality sensor, the website “WebPlotDigitizer” [3] was used to calculate the sensor sensitivity characteristic points for CO<sub>2</sub> by figure 20. Extracting the acquired data from the graph (figure 20), reference [4] website was used in order to calculate the value of “A” and “B” constants that were used for the process of

calculation of ppm in the code [4]. After that the values that were extracted from the graph were placed in excel and the correlation between ppm and Rs/Ro graph for CO<sub>2</sub> was created.

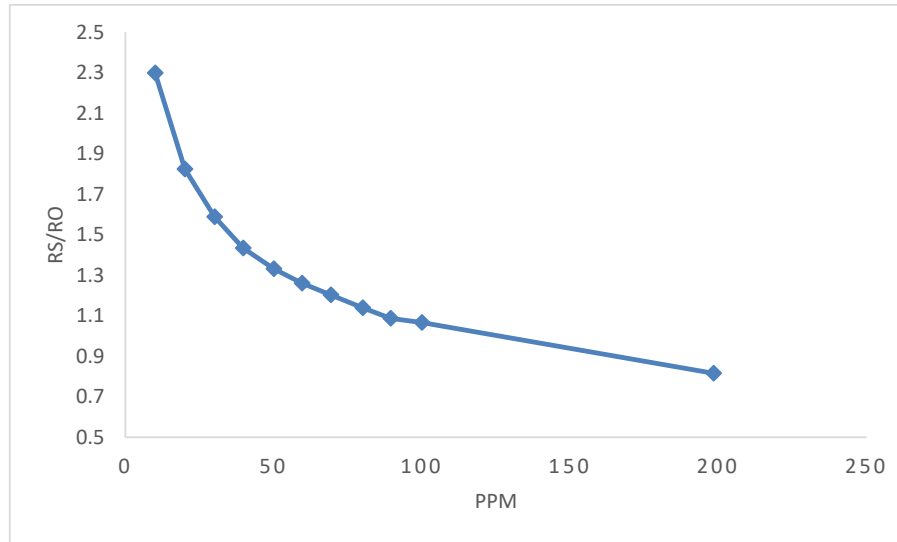


Figure 21: PPM & Rs/Ro correlation graph for CO<sub>2</sub>

Finally, the correlation is mathematically modelled using the equations 2-4.

$$Rs = ((1024.0 * R_L) / avgRo) - R_L \quad (2)$$

$$RR = Rs / Ro \quad (3)$$

$$ppm = (a * pow(RR, b)) * 1000 \quad (4)$$

Unfortunately, we did not have any means of verifying the measured CO<sub>2</sub> valued. **The Air Quality sensor integration with the existing microcontroller was the requirements as indicated in the KPIs.**

#### 1.4.2 References

- [1] Q. components, "MQ-135 Air Quality Gas Sensor Module," 2022. [Online]. Available: <https://quartzcomponents.com/products/mq-135-air-quality-gas-sensor-module> [Accessed 14 5 2023].
- [2] H. E. C. LTD, "TECHNICAL DATA MQ-135 GAS SENSOR," 2018. [Online]. Available: [https://www.electronicoscaldas.com/datasheet/MQ-135\\_Hanwei.pdf](https://www.electronicoscaldas.com/datasheet/MQ-135_Hanwei.pdf) [Accessed 14 5 2023].
- [3] WebPlotDigitizer, "WebPlotDigitizer," 2021. [Online]. Available: <https://automeris.io/WebPlotDigitizer/> [Accessed 14 5 2023].
- [4] k. O. Calculator, "Power regression Calculator," 2022. [Online]. Available: <https://keisan.casio.com/exec/system/14059931777261> [Accessed 14 5 2023].

## 1.5 Light Intensity Sensor – Custom made Using GL5516 LDR

For the purposes of this project the GL5516 LDR (Light Dependant Resistor) sensor was used. The GL5516 LDR sensor is a light dependant resistor which is constructed from a semiconductor materials and the conductivity changes based on the intensity of the light, called photoconductivity. A LDR sensor can be used in light-sensitive detector circuits and light and dark activated switching circuits. The sensor's size is 5mm x 2mm and is functioning with 5V.

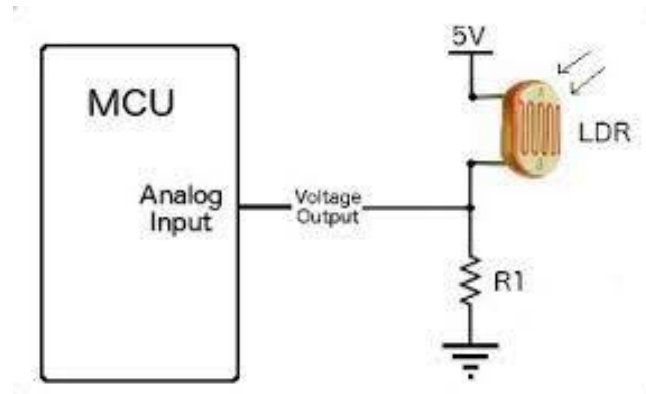


Figure 22: LDR in voltage divider configuration

The GL5516 LDR sensor is connected to the microcontroller in voltage divider configuration, as shown on figure 22. The voltage that appears at the analogue input (A1) will vary depending on the amount of light intensity that is hitting the sensor. The LDR is connected to 5V. A pull-down resistor is required for the wiring of the sensor. A 10 kΩ resistor (R1) was chosen for the purposes of this project.

### 1.5.1 Light Intensity Sensor Integration

The integration of the custom made light intensity sensor is represented by Figure 23. If the result is smaller or equal to 400, then a “Night” sign appears on the screen and led is activated (“Led”). If this statements is not true then a “Day” sign appears on the screen and led is disable (“Led”).

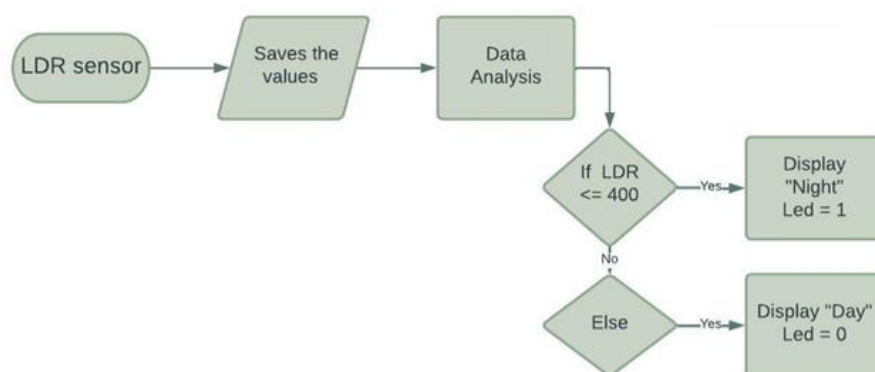


Figure 23: LDR Seneor Flowchart

### 1.5.2 *Light Intensity Sensor Calibration*

The light intensity sensor was mathematically modelled as follows:

$$V_{sensor} = V_{uc} = \frac{(10k)(3.3)}{10k + R_{LDR}} \quad (5)$$

Where  $V_{sensor}$  is the voltage drop across the 10k Ohm resistor which is also the same voltage drop that the microcontroller ( $V_{uc}$ ) is reading as an input. Then using (5) we can solve for the resistance of the LDR sensor ( $R_{LDR}$ )

$$R_{LDR} = \frac{(10k)(V_{uc})}{3.3 - V_{uc}} \quad (6)$$

Figure 24 shown an example of light intensity sensor, where it is detected that it is night (reduced light intensity) and the system also suggests or automatically can proceed to turn on the lights thus optimizing the crop production yield



*Figure 24: Example of Light Intensity Sensor Output*

**The successfully development and integration of Light Intensity Sensor satisfies the set KPI requirement.**

## 1.6 Soil Moisture Sensor – Custom made

Soil moisture sensors estimate volumetric water content in the soil. Most of them are stationary and are placed in predetermined locations and depths in the field. By using some other property of the soil, such as electrical resistance, dielectric constant, or interaction with neutrons, as a proxy for the moisture content the sensor measures the soil moisture in the field. In this project the estimation of the soil moisture is done by the measurement of the electrical resistance of the soil.

The development of a cheap and robust soil humidity sensor started with the concept of a voltage divider with two resistors. One resistor had a fixed value of 5.6k Ohms and the other resistor would be the soil resistance which changes with the amount of water content. The basic setup is shown on Figure 25. The mathematical modelling is no different that the equation already presented, hence will be omitted. However, the extensive experimentation procedure and results follow.

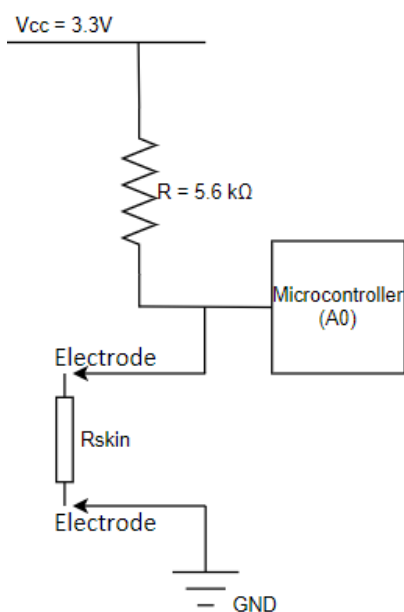


Figure 25: Soil Moisture Circuit

The actual experimental setup is shown on figure 26. A plastic pot was filled with soil. The volume of the added soil was calculated and measured. Then the soil humidity was emulated by adding the equivalent amount of water.



Figure 26: Soil Humidity Experimental Setup

The soil resistance was measured across the probes that were inserted in the soil at different distances.

### 1.6.1 Soil Humidity Sensor Integration

Figure 27 showcases the flowchart of the soil moisture sensor that explains the path of the code. If the values of “X1” and “X2” are collectively larger than zero, then their average value is calculated and a “Soil moisture:” sign appears on the screen demonstrating the average value of soil moisture. If “X1” value is smaller than zero and “X2” is larger than zero, then a “Soil moisture:” sign appears on the screen demonstrating the “X2” value. And finally if the “X1” value is larger than zero while the “X2” value is smaller than zero, will result to a “Soil moisture:” sign appearing on the screen demonstrating the “X1” value (figure 28).

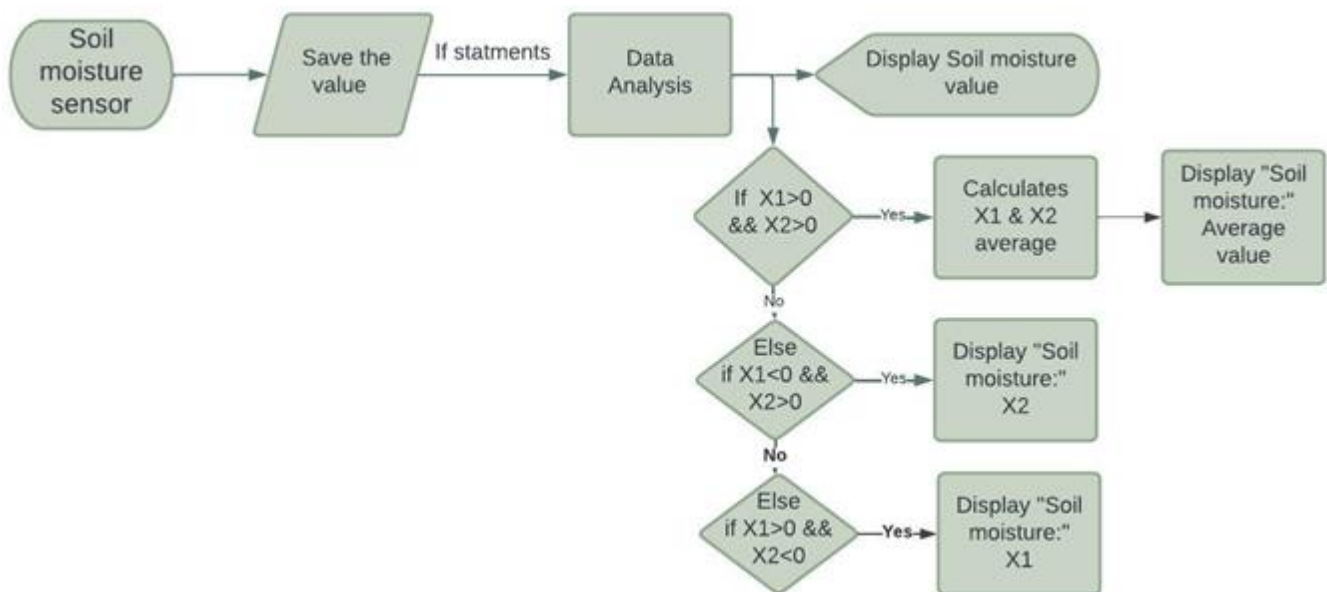


Figure 27: Soil Humidity Sensor Integration Flowchart



Figure 28: Soil moisture displayed on the uLCD screen

### 1.6.2 Soil Humidity Sensor Calibration

In order to select the correct value of resistor for the voltage divider circuit that will measure the electrical resistance of the soil, a variety of fixed resistors (1.5k $\Omega$ , 5.6k $\Omega$ , 10k $\Omega$ , and 1.5M $\Omega$ ) were tested with a diversity of resistances ranging from 1k $\Omega$  to 1M $\Omega$ , in order to find the percentage error (%) of each fixed resistor (table 3).

Table 3: Voltage divider resistor error range

| Real Values ( $\Omega$ ) | Fixed Resistors ( $\Omega$ ) |         |         |         |
|--------------------------|------------------------------|---------|---------|---------|
|                          | 1.5k                         | 5.6k    | 10k     | 1.5M    |
| 1k                       | 1.740                        | -1.790  | -0.380  | 895.520 |
| 1.5k                     | -0.147                       | -1.773  | -0.453  | 737.227 |
| 10k                      | 0.244                        | -0.634  | 0.735   | 331.176 |
| 47k                      | 0.532                        | -0.773  | -2.087  | 363.194 |
| 56k                      | -4.744                       | 0.081   | -2.135  | 334.897 |
| 68k                      | -8.110                       | -1.515  | -4.209  | 297.313 |
| 100k                     | 2.611                        | -1.835  | -6.851  | 297.313 |
| 220k                     | -4.404                       | -2.318  | -15.485 | 117.408 |
| 560k                     | -8.859                       | -15.687 | -33.445 | 46.058  |
| 1M                       | 53.423                       | -12.360 | -47.818 | 18.114  |
|                          | Error range (%)              |         |         |         |

After the calculation of the percentage error of each fixed resistor, the error of the 1.5k $\Omega$  resistor was found to be 23%, the 10k $\Omega$  was 40%, the 1.5M $\Omega$  was 78% and the 5.6k $\Omega$  was found to have the smallest error range of 11%. Analytic information is given in Figure 29 which showcases the error range values of every fixed resistor that was taken through the voltage divider circuit for each resistor.

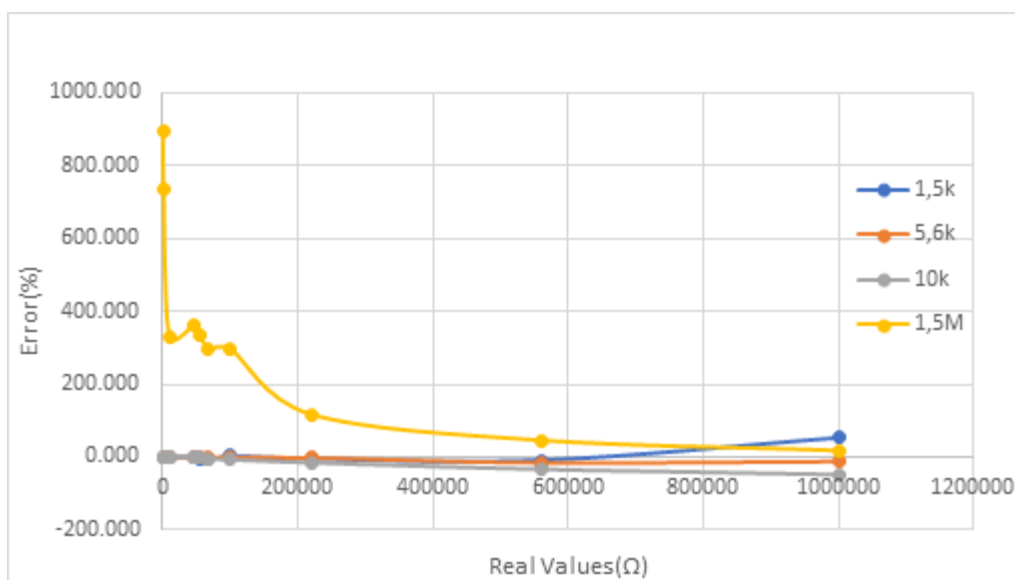


Figure 29: Error range of each resistor

After the selection of the 5.6 kΩ resistor, a plethora of measurements were done, in order to find the resistivity of the soil in different amounts of water content. A flow pot with a known volume of soil was selected and the probes were installed inside in a predetermined depth (9 cm and 6 cm). The purpose of the experiment was to measure the resistivity of the soil after the watering of the soil in a fixed amount of water. Table 4 showcases the measurements that were taken from the experiment that took place (table 4).

Table 4: Resistivity of the soil in different water content

| Probes | Water Content | Ratio of Water to Soil Volume |       |       |       |       |       |       |      |      |       |
|--------|---------------|-------------------------------|-------|-------|-------|-------|-------|-------|------|------|-------|
|        | 0%            | 10%                           | 20%   | 30%   | 40%   | 50%   | 60%   | 70%   | 80%  | 90%  | 100%  |
|        | (Ω)           | (Ω)                           | (Ω)   | (Ω)   | (Ω)   | (Ω)   | (Ω)   | (Ω)   | (Ω)  | (Ω)  | (Ω)   |
| 1 -> 2 | 25500         | 16500                         | 17000 | 14500 | 13500 | 4300  | 4000  | 3800  | 3600 | 3500 | 3400  |
| 1 -> 3 | 24500         | 13000                         | 13250 | 11500 | 11000 | 4400  | 4300  | 4100  | 4300 | 4000 | 3800  |
| 1 -> 4 | 25900         | 14000                         | 12750 | 6000  | 5300  | 4600  | 4800  | 4700  | 4500 | 4400 | 4200  |
| 2 -> 3 | 40000         | 15000                         | 13000 | 5600  | 5500  | 12000 | 11000 | 10000 | 9000 | 8800 | 10000 |
| 2 -> 4 | 44000         | 14300                         | 12000 | 5800  | 5000  | 12000 | 11000 | 10000 | 8900 | 8600 | 8500  |
| 3 -> 4 | 32000         | 14800                         | 13000 | 5500  | 5200  | 5000  | 5000  | 5200  | 5000 | 4800 | 4500  |

After the end of the experiment and the collection of the measurements were graphed as and presented on Figure 30 for better visualization. Right away it can be seen from figure 30 that most measurements have a trend, where soil resistance is inversely proportional to the water content. This relationship is logical and makes too much sense because the majority of resistive sensors have similar relationship with water. However, from figure 30 we can see that some



measurements between probes 2-3 and 2-4 do not follow the same relationship. Hence those measurements are disregarded and shown on Figure 31.

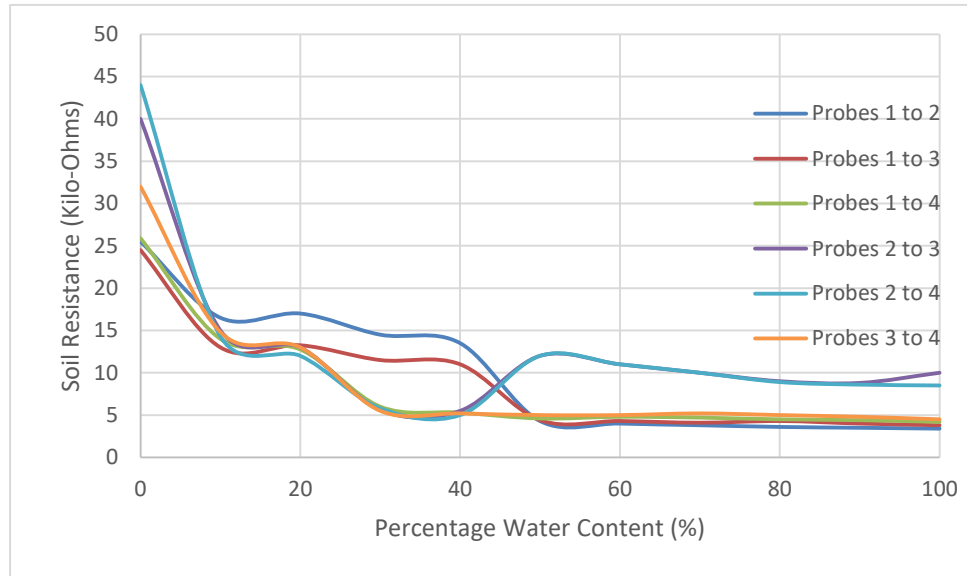


Figure 30: Soil Resistance Experimental Data

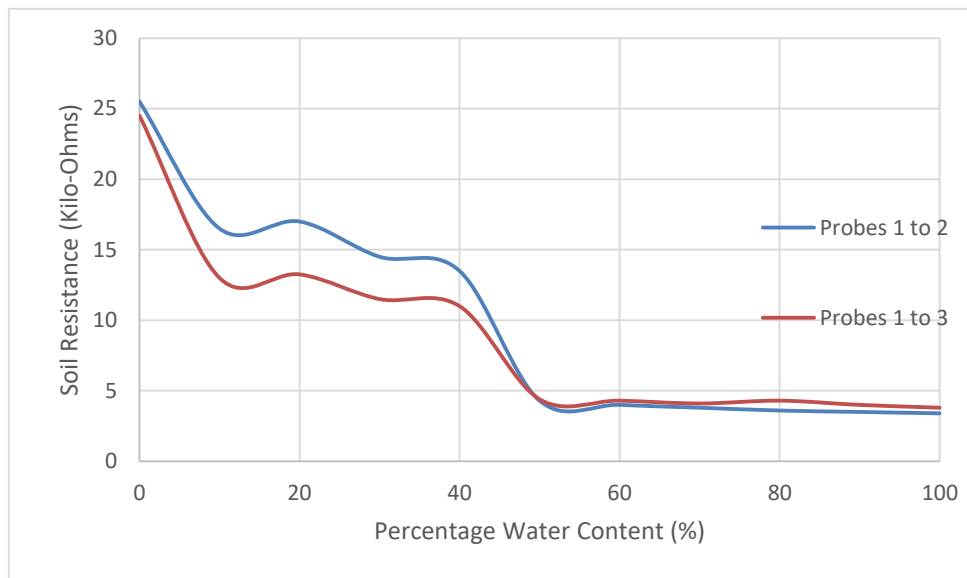


Figure 31: Results for Probes 1-2 and 1-3

Taking a closer look at the results, after rejecting the results between probes 2-3 and 2-4, it was found are clearly shown on figures 31 and 32 that the best results are between probes 1-3 and 3-4. When the probes are connected between 1-2 and 1-3 the percentage difference of measured resistance for soil humidity less than 40% is high.

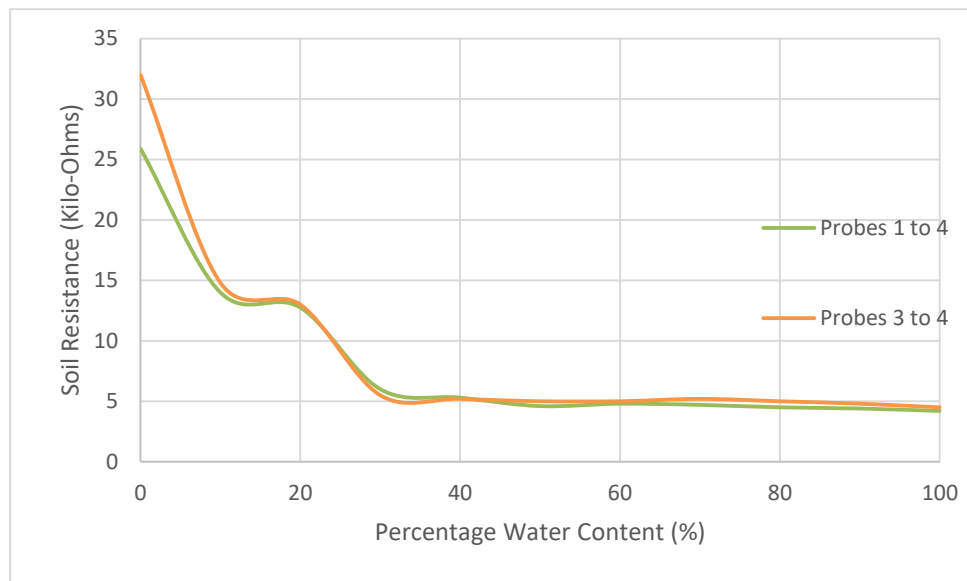


Figure 32x: Results for Probes 1-4 and 3-4

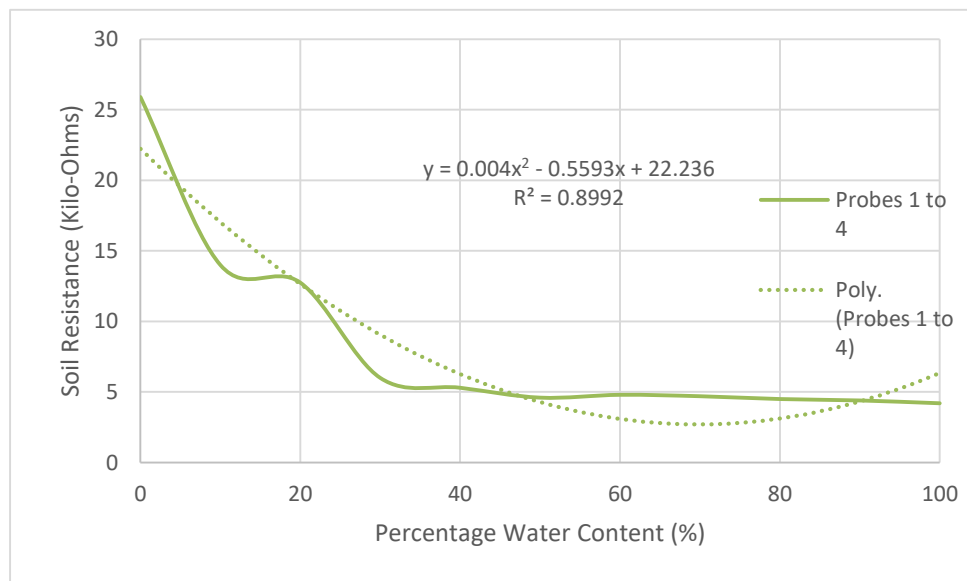


Figure 33: Data Analysis using 2nd Order Polynomial

Therefore, it can be concluded to proceed with probe connection between 1 and 4. Data analysis show that the relationship between soil resistance and soil humidity when fitted on a 2<sup>nd</sup> order polynomial has an accuracy of 90% whereas a 3<sup>rd</sup> order increases the accuracy to 97%.

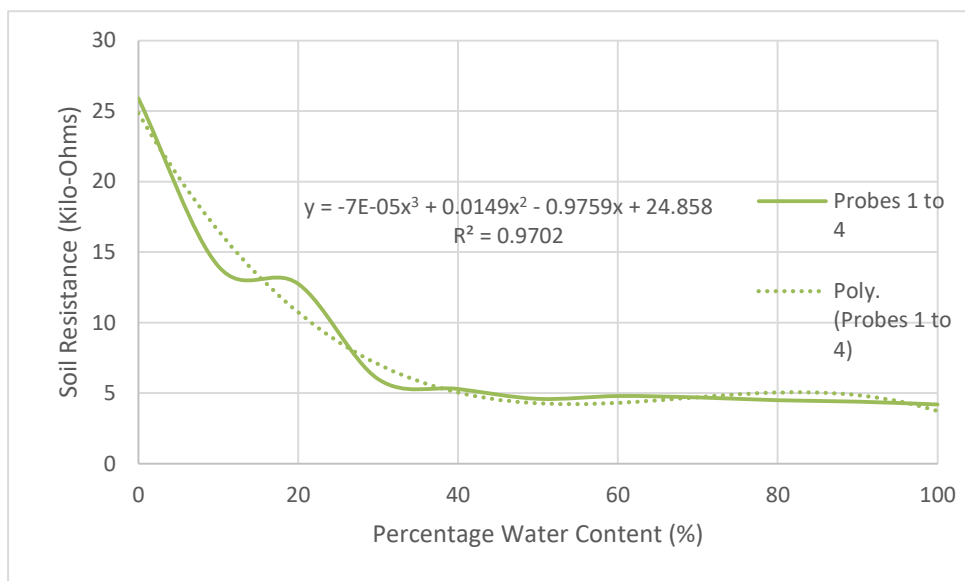


Figure 34: Data Analysis using 3rd Order Polynomial

However, everything in engineering can be considered as an optimization problem having multiple solutions and constraints and the engineer needs to select the best possible solution which best fits the project requirements. High accuracy is better, however the computing complexity and power required when implementing a 3<sup>rd</sup> order polynomial on a microcontroller increases exponentially. For this reason it was concluded that the 2<sup>nd</sup> order polynomial would be the best solution for the microcontroller proposed solution. **The soil humidity accuracy of 90% meets the requirements as indicated in the KPIs.**

## 1.7 Monitoring System Output Examples

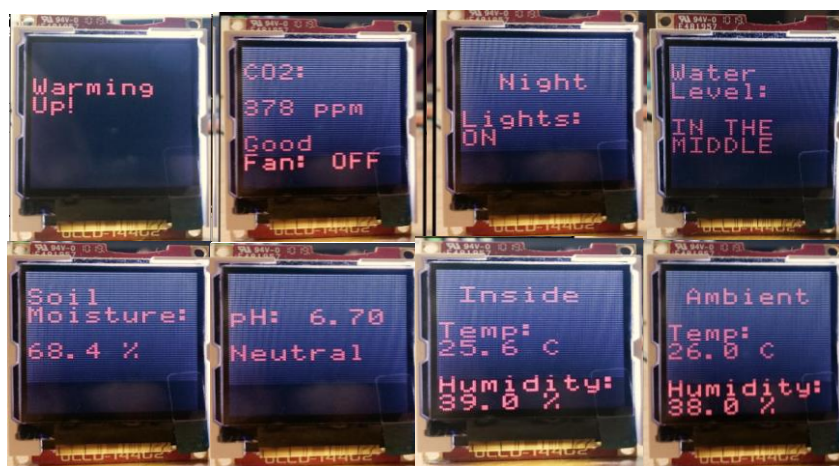


Figure 35: Monitoring System Output Examples

## 2 Image Processing using YOLO

After an extensive literature review on image processing algorithms for small UAV applications as presented on Deliverable 3.1, it was concluded to proceed with the YOLO algorithm. Furthermore it was decided first to proceed and test the capabilities for Fire Detection because of abundance of datasets compared to insect detection in Cyprus.

Convolutional neural networks (CNNs) are one of the representative algorithms of deep learning and are currently achieving the best results in areas such as target detection and classification. However, these networks usually require graphics processing units to operate properly because they are computationally challenging, use large amounts of memory, and are expensive to run. Therefore, deploying real-time target detection algorithms on devices without powerful computing power or on embedded devices is one of the current challenges in computer vision.

In this case, a target detection system that could easily and flexibly replace different models, hardware, and thus achieve applicability to different scenarios based on different usage requirements would have a significant impact on the large-scale deployment of target detection algorithms.

### 2.1 Fire Detection

With the explosion of technology in the field of computer vision in recent years, target detection algorithms have begun to appear in a variety of areas of need. These algorithms are often used in emergency situations that require fast and precise response. One example is in fire detection, as fires are mostly associated with harmful effects such as economic loss and destruction of natural resources. In such emergency situations, any small delay can cause irreparable damage. A target detection system that is effective, dependable, and simple to deploy in various conditions is required in this case.

We tried to deploy the SoTA target detection algorithm on a low computing power embedded device (Raspberry Pi 4B) to detect forest fires in real time for small UAV applications:

- We tried to modify the original YOLOv5 backbone to a lightweight network, and restructured the whole network based on the new backbone.
- We performed a channel pruning operation on the modified YOLO network to make the network further compact and the network structure more simplified.
- We overclocked the CPU of Raspberry Pi 4B at the hardware level and investigated the effect of hardware acceleration on detection frame rate.

Starting from the comparison between YOLO versions as presented on Figure 36, benchmark experimental results show that the proposed YOLOv5 has a higher accuracy rate (mAP@0.5) than the original YOLOv5 on the same test set, and that the detection frame rate(FPS) has been significantly improved(1.16 FPS to 8.57 FPS).

| Model         | size          | objects | mAP  | Jetson Nano 1479 MHz | RPi 4 64-OS 1950 MHz |
|---------------|---------------|---------|------|----------------------|----------------------|
| NanoDet       | 320x320       | 80      | 20.6 | 26.2 FPS             | 13.0 FPS             |
| NanoDet Plus  | 416x416       | 80      | 30.4 | 18.5 FPS             | 5.0 FPS              |
| YoloFastestV2 | 352x352       | 80      | 24.1 | 38.4 FPS             | 18.8 FPS             |
| YoloV2        | 416x416       | 20      | 19.2 | 10.1 FPS             | 3.0 FPS              |
| YoloV3        | 352x352 tiny  | 20      | 16.6 | 17.7 FPS             | 4.4 FPS              |
| YoloV4        | 416x416 tiny  | 80      | 21.7 | 16.1 FPS             | 3.4 FPS              |
| YoloV4        | 608x608 full  | 80      | 45.3 | 1.3 FPS              | 0.2 FPS              |
| YoloV5        | 640x640 small | 80      | 22.5 | 5.0 FPS              | 1.6 FPS              |
| YoloV6        | 640x640 nano  | 80      | 35.0 | 10.5 FPS             | 2.7 FPS              |
| YoloV7        | 640x640 tiny  | 80      | 38.7 | 8.5 FPS              | 2.1 FPS              |
| YoloX         | 416x416 nano  | 80      | 25.8 | 22.6 FPS             | 7.0 FPS              |
| YoloX         | 416x416 tiny  | 80      | 32.8 | 11.35 FPS            | 2.8 FPS              |
| YoloX         | 640x640 small | 80      | 40.5 | 3.65 FPS             | 0.9 FPS              |

Figure 36: YOLO Performance Benchmark

After carried out a research, the mainstream single board computer/embedded platform for deploying deep learning algorithms are Raspberry Pi and Nvidia Jetson series. NVIDIA's Jetson series is seen as a deployment accelerator for machine deployment. Some deep training applications of Jetson Nano developers are better evaluated than Raspberry Pi kit.

On the other hand, when comparing the electrical power consumption as presented on Figure 37, it can be seen that Raspberry Pi consumes 25% less energy than Jetson nano. In addition, Raspberry Pi is much lighter (46 grams) compared to NVidia Jetson nano which weighs at 241 grams.

Therefore, for small UAV applications it was decided to proceed with Raspberry Pi Single board computer.

|                  | Raspberry PI 4                           | Jetson Nano                                |
|------------------|--|--|
| Performance      | 13.5 GFLOPS                              | 472 GFLOPS                                 |
| CPU              | Quad-core ARM CortexA72 64-bit @ 1.5 GHz | Quad-Core ARM Cortex-A57 64-bit @ 1.42 GHz |
| GPU              | Broadcom Video Core VI (32-bit)          | NVIDIA Maxwell w/ 128 CUDA cores @ 921 MHz |
| Memory           | 8 GB LPDDR4                              | 4 GB LPDDR4 @ 1600MHz, 25.6 GB/s           |
| Networking       | Gigabit Ethernet / WiFi 802.11ac         | Gigabit Ethernet / M.2 Key E               |
| Display          | 2x microHDMI (up to 4Kp60)               | HDMI 2.0 and eDP 1.4                       |
| USB              | 2x USB 3.0, 2x USB 2.0                   | 4x USB 3.0, USB 2.0 Micro-B                |
| Other            | 40-pin GPIO                              | 40-pin GPIO                                |
| Video Encode     | H264(1080p30)                            | H.264/H.265 (4Kp30)                        |
| Video Decode     | H.265(4Kp60) ,H.264(1080p 60)            | H.264/H.265 (4Kp60, 2x 4Kp30)              |
| Camera           | MIPI CSI port                            | MIPI CSI port                              |
| Storage          | Micro-SD                                 | 16 GB eMMC                                 |
| Power under load | 2.56W-7.30W                              | 5W-10W                                     |

Figure 37: Comparison between RPi4B and Jetson nano

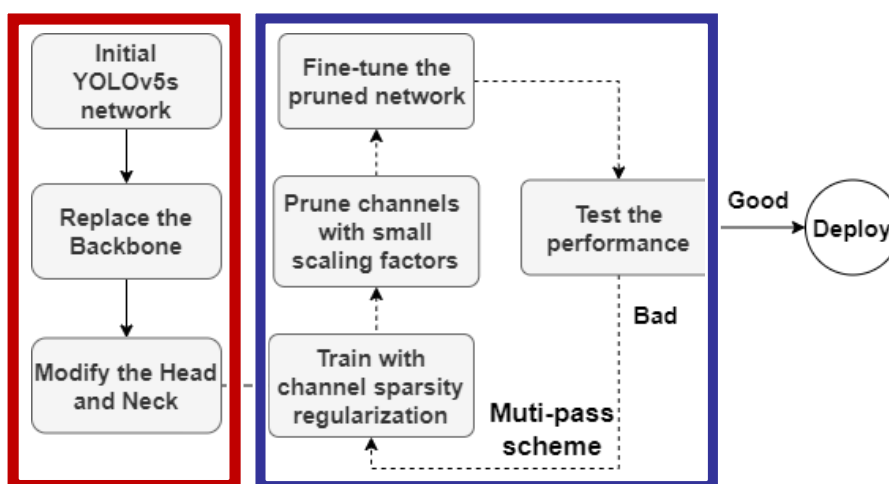


Figure 38L Proposed optimization Framework

The proposed optimization framework as presented on Figure 38, proposes to start with the initial YOLOV5s and then replace the backbone, followed by modification of the network's head and neck.

The network architecture is presented on Figure 39. The architecture of the proposed network. 1) The input is a  $320 \times 320$  three-channel RGB image. 2) The backbone of the proposed network is ShuffleNetV2, which can reduce the

amount of cache space occupied and increases the inference speed. 3) The Neck network part uses a FPN + PAN architecture, with channel pruning of the Head in order to optimise memory access and usage.

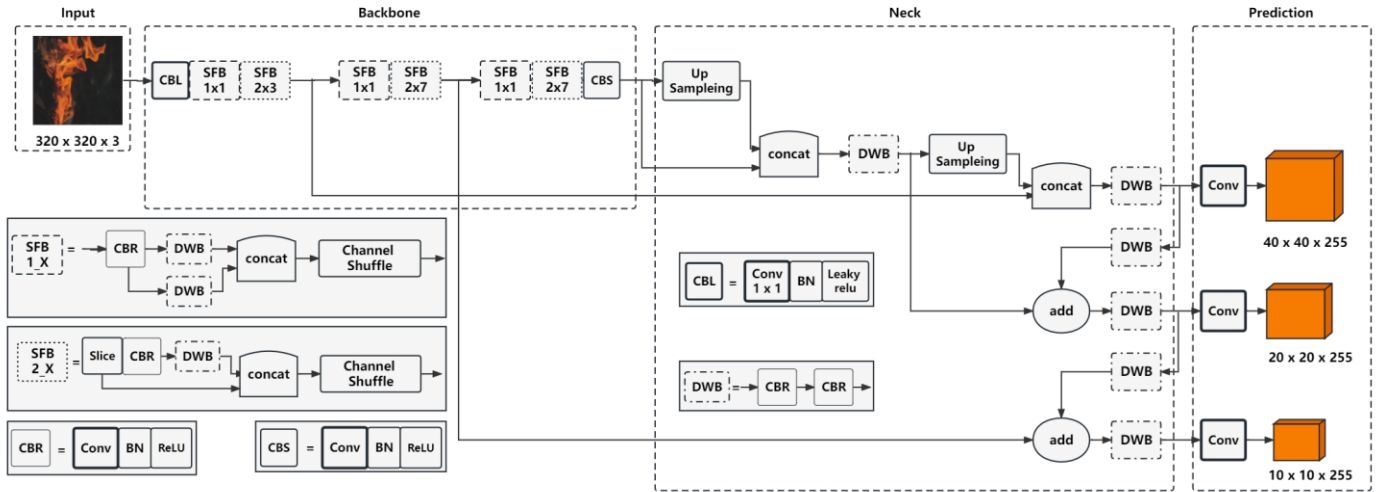


Figure 39: Network Architecture

Figure 40: represents the Network Pruning. We associate a scaling factor (reused from batch normalization layers) with each channel in convolutional layers. Sparsity regularization is imposed on these scaling factors during training to automatically identify unimportant channels. The channels with small scaling factor values (in orange color) will be pruned (left side). After pruning, we obtain compact models (right side), which are then fine-tuned to achieve comparable (or even higher) accuracy as normally trained full network.

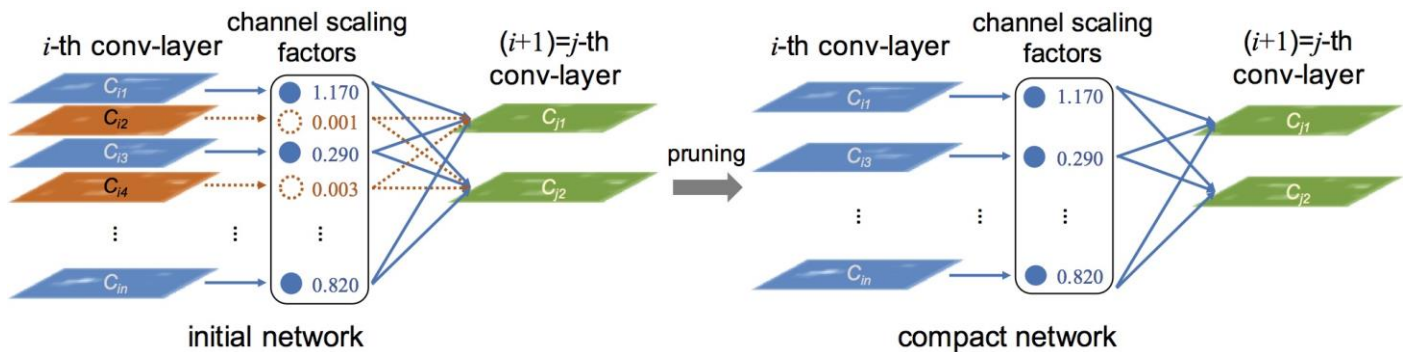


Figure 40: Network Pruning

Sparse training and pruning preparation is shown on Figure 41. It uses the scaling factors of the BN layer and associates the scaling factors with each channel in the convolutional layer. A sparse regularization is applied to these scale factors during training so that the unimportant scale factor is approximated to zero, thus automatically identifying the unimportant channels.



By pruning the orange channels with a scale factor close to 0, we can obtain the compact network, which is the channel with a larger scale factor.

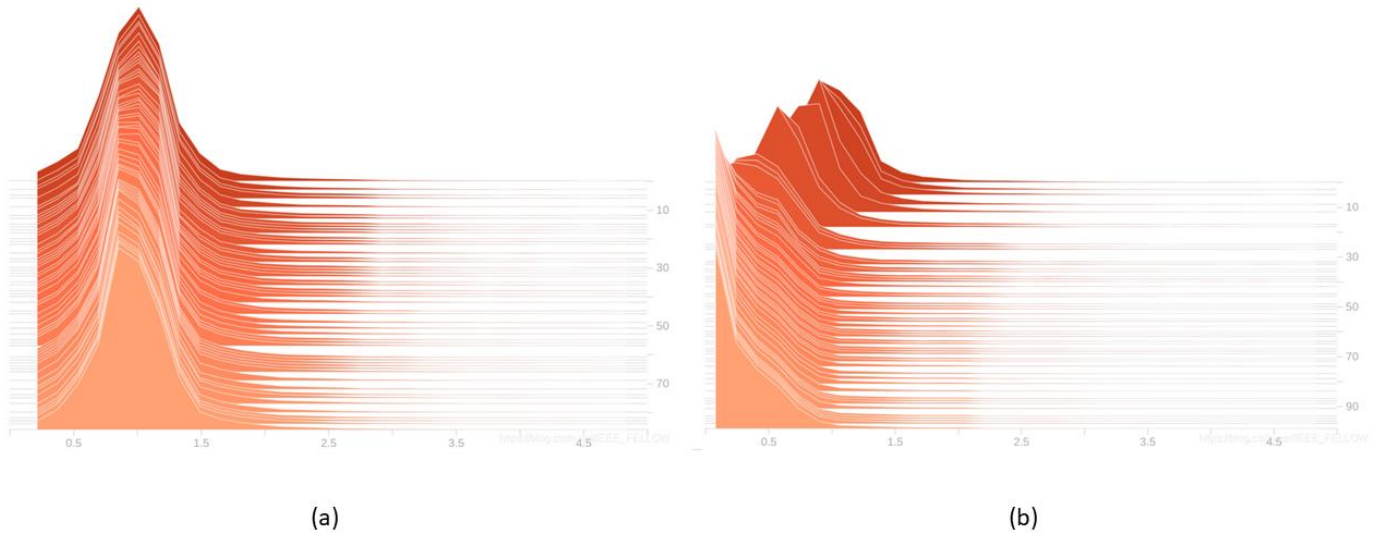


Figure 41: Sparse training and pruning preparation

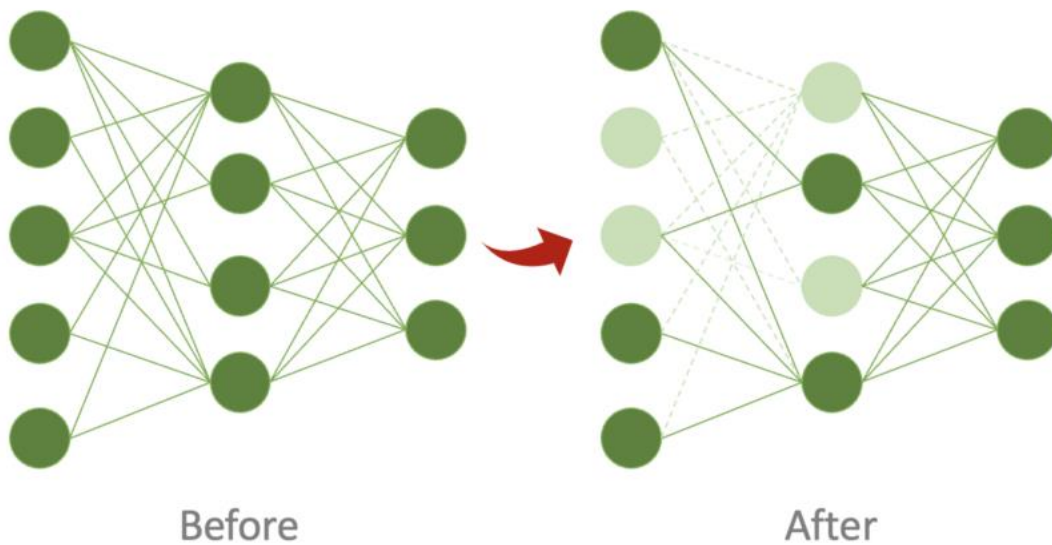


Figure 42: Pruning process and Fine-tuning of the pruned model

The Pruning process and Fine-tuning of the pruned model is visualised on Figure 42. After obtaining the sparse trained model, the next step is to prune out the channels with  $\gamma$  going to 0. This method is based on the structured pruning of the channels, and the accuracy of the pruning will generally be reduced. We can fine-tune the finetune of the compact network to improve its accuracy, so that it is comparable to the normal training network, or even more accurate.

The reduction in the accuracy after pruning can be fine-tuned to recover. When the pruned model is able to relearn the neural network parameters based on the current network structure, it is the fine-tuned for training and this restores the detection accuracy of the model and improves the mapping effect.



| Clock (MHz) | Overvoltage | V <sub>core</sub> | Max temp. (°C   °F) | Power (Watt) | Performance increase | Remarks                 |
|-------------|-------------|-------------------|---------------------|--------------|----------------------|-------------------------|
| 0           | 0           | 0.8625            |                     | 1.5          |                      | RPi 4 shut down         |
| 200         | 0           | 0.8625            |                     | 1.75         |                      | RPi 4 min working clock |
| 600         | 0           | 0.8625            |                     | 2.8          |                      | RPi 4 running idle      |
| 1500        | 0           | 0.8625            | 82   180            | 7            |                      | Factory settings        |
| 1600        | 1           | 0.8875            | 80   176            | 7.6          | 6.6 %                |                         |
| 1700        | 2           | 0.9125            | 78   172            | 8.3          | 13.3 %               |                         |
| 1800        | 3           | 0.9375            | 77   170            | 8.9          | 20 %                 |                         |
| 1900        | 4           | 0.9625            | 75   167            | 9.5          | 26.6 %               |                         |
| 2000        | 6           | 1.0125            | 72   162            | 11           | 33.3 %               |                         |
| 2100        | 6           | 1.0125            | 72   162            | 11           | 40 %                 |                         |
|             | 7           | 1.0375            | 56   132            | 11.7         |                      | no improvement          |
|             | 8           | 1.0625            | 50   122            | 12.3         |                      | no improvement          |

Figure 43: Hardware Acceleration Results

The hardware acceleration results are tabulated on Figure 43. In order to get the best performance out of the algorithms, the RPi4B CPU was overclocked to 2.0 GHz (maximum of 2140 MHz). As the RPi is normally used with a CPU with its NEON-ARM instructions, the GPU was not overclocked for this study and its default frequency, 500 MHz was used (maximum of 650 MHz). To avoid overheating of the RPi platform, automatic over-voltage adjustment and dynamic clock frequency were used.

Some examples of the dataset is presented on Figure 44. The dataset includes scenarios that look like fire but in reality they are false positives. Figure 44, (a) and (b) show image-based data augmentation techniques to expand the dataset. (c) and (d) show respective ground-truth bounding boxes. (e) and (f) represent images that are prone to false detection. (g) is a ground-based image of the forest fire and (h) is an aerial view of the fire.

The dataset was randomly divided into three independent and equally distributed sets:

- (i) the Training set, containing 83% of the images (20,255);
- (ii) the Validation set, containing 13% of the images (3,148);
- (iii) the Test set, containing 4% of the images (977).



Figure 44: Fire Detection Dataset

As presented on Figure 45, the optimised YOLOv5 outperforms the original YOLOv5 across most of the evaluation metrics. However, the location of objects detected by the original YOLOv5 network is on average more accurate in terms of IoU experimental values. This is because the original YOLOv5 retains more convolutional layers than the optimised network.

| Network  | mAP@0.50<br>(%) | AP@0.50<br>smoke (%) | AP@0.50<br>fire (%) | $F_1$<br>score | avg IoU<br>(%) |
|----------|-----------------|----------------------|---------------------|----------------|----------------|
| YOLOv5   | 82.93           | 91.17                | 74.86               | 0.83           | 73.51          |
| Proposed | 92.54           | 96.35                | 88.71               | 0.85           | 68.34          |

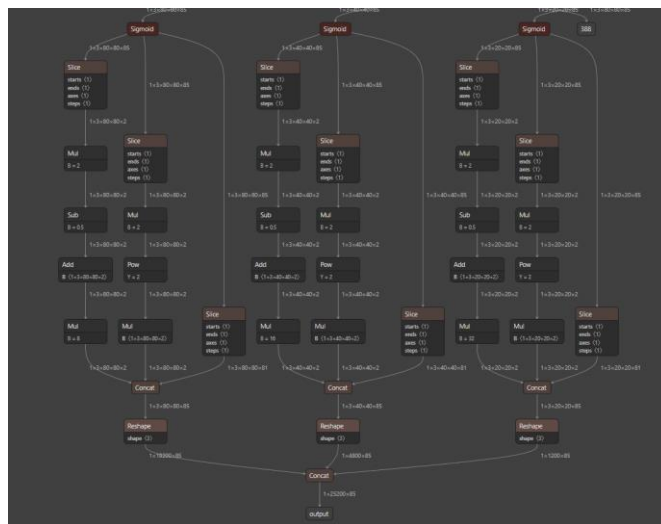
Figure 45: Comparison of models 'performances on Prediction

Parameter comparison of the proposed detection models for different channel pruning rates are tabulated on Tab. II, According to Tab. II, all 4-evaluation metrics were reduced for different channel pruning rates. After performing fine-tuning training, the mAP recovered to 84.87%, 92.5%, 92.65% and 93.41% respectively.

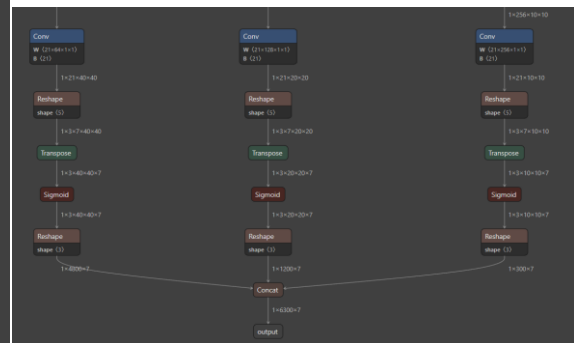
| Pruning Rate(%) | mAP@0.5(%) | Parameters/ $10^6$ | Model Size(MB) |
|-----------------|------------|--------------------|----------------|
| Baseline        | 93.7       | 7.02               | 14.1           |
| 80              | 82.3       | 0.89               | 1.95           |
| 70              | 89.7       | 1.54               | 3.34           |
| 60              | 91.2       | 2.30               | 4.64           |
| 50              | 93.2       | 2.85               | 5.66           |

Figure 46: Analysis of the model pruning results

Fine-tuning revealed that the channel pruning rate of 70% achieved the best balance between accuracy and inference speed, which resulted in a better model compression with less loss of average accuracy.



Original Yolov5 Head



Optimized Yolov5 Head

Figure 47: Network Structure Comparison

As it is clearly presented on Figure 47, the proposed optimised Yolo V5 has considerable smaller head without compromising any performance.

Comparison results on the performance metrics as presented on Figure 48. As presented the proposed algorithm increases the inference speed by a factor of 7.4 at all clock speeds, reaching a highest value of 8.57 FPS at 2GHz CPU clock speed. In addition, the proposed network achieves a reduction of the CPU usage in the range of 35% (from high 90s to low 60s) which also translates in a 25% reduction of CPU temperatures. Compared to the state-of-the-art literature presented in previous sections, this work increases inference speed by 50% and reduces the CPU usage by 35%. Finally, the proposed work achieves a 10% reduction in electrical power consumption which prolongs operation run-time and range of applications involving small UAVs. As research shows the battery run-time is not linear but exponential.

| Network  | Clock Speed<br>(GHz) | Power<br>(W) | CPU Usage<br>(%) | CPU Temp<br>(°C) | FPS  |
|----------|----------------------|--------------|------------------|------------------|------|
| YOLOv5   | 0.6                  | 5.70         | 93               | 47               | 0.41 |
| YOLOv5   | 1.8                  | 7.70         | 95               | 58               | 1.06 |
| YOLOv5   | 2.0                  | 8.10         | 97               | 67               | 1.16 |
| Proposed | 0.6                  | 5.20         | 60               | 43               | 3.02 |
| Proposed | 1.8                  | 7.02         | 58               | 47               | 7.23 |
| Proposed | 2.0                  | 7.28         | 64               | 50               | 8.57 |

Figure 48: Overclocking Performance Comparison

The detection results (Figure 49) show that the proposed YOLOv5 can mark out more areas where flames and smoke are present in the pictures and there are no false detections in pictures with flame-like objects.

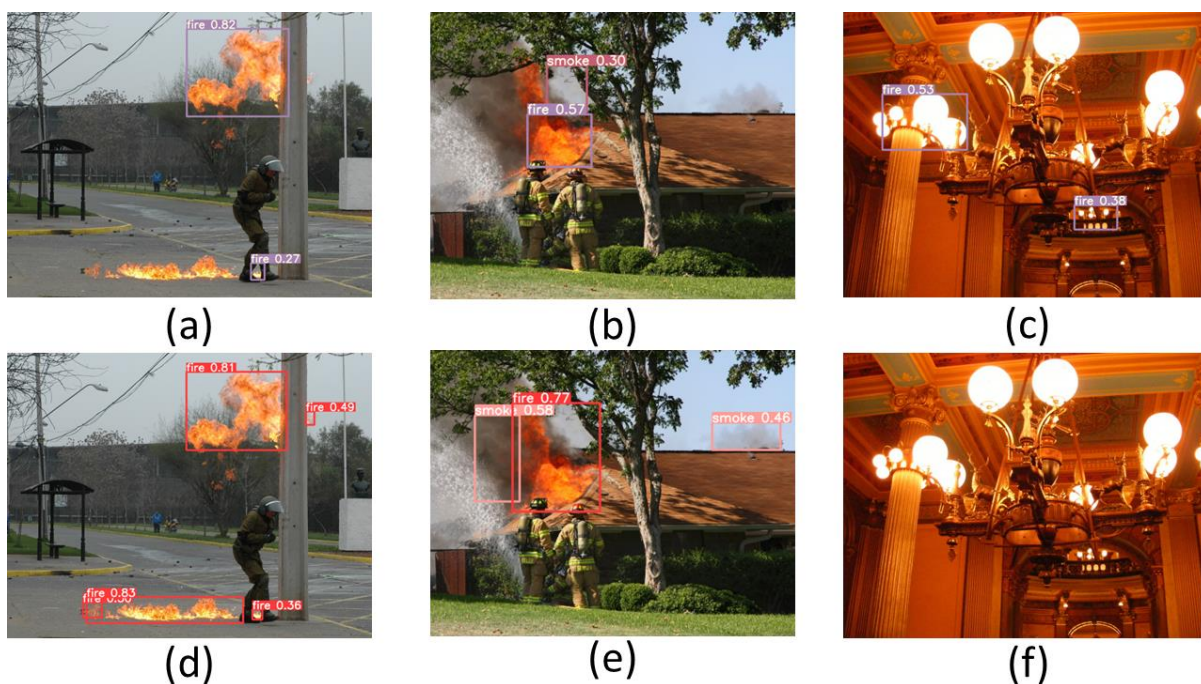


Figure 49: Detection results from the Comparison Experiment

Evaluation on the Raspberry Pi platform are presented on Figure 50. The detection result of the image obtained in real time from the web camera, with the live detection frame rate in the top left corner of the detection screen.





Figure 50: Detection Results on Raspberry Pi Platform



Figure 51: Video input Results for testing the model performance

A video of the California fires was obtained from the internet and the results of flame and smoke detection using the optimized model show that the model can still detect the location of the flames even in a smoke-filled scene and mark the location of the flames when the camera is zoomed out.

Conclusions which satisfy the set KPIs.

- The optimized YOLOv5 produced the highest mAP of 92.5% compared to the ordinary YOLOv5s model and could detect at 7-9 FPS on the RPi-4B.
- The optimized model use 35% less CPU usage than the original YOLOv5.
- The reduced CPU usage also translated to 25% reduction in CPU temperature.
- The deployment approach in this study reduces the difficulty of deploying the deep-learning fire detection model on edge devices.

## 2.2 Sensitivity Analysis

This proposal presents a system architecture that uses deep learning image processing techniques to learn features for detecting desired targets and train neural network models to automatically identify targets. The technology can automatically, quickly and accurately identify targets in scenes with surveillance cameras and set up early warning devices to respond in a timely manner, gaining more time and minimizing damage in the early stages of an incident.

In contrast to the traditional algorithm development process, this proposal takes a novel approach in which the algorithm determines the hardware. A set of highly robust algorithms is first determined, and then the appropriate hardware is selected according to the requirements of various usage scenarios to achieve the desired results. Compared with the traditional target detection system where one set of hardware is bound to only one set of algorithms, the target detection system of this architecture can make the most suitable adjustments according to different usage scenarios and requirements. In terms of flexibility, this new development process will be more advantageous.

The proposal uses a lightweight target detection algorithm based on the improved YoloV5. YOLO (You Only Look Once) is one of the most accurate and precise object detection algorithms available. The 3 main image processing algorithms currently available are Single Shot MultiBox Detector (SSD), Faster Region based Convolutional Neural Networks (Faster R-CNN) and You Only Look Once (YOLO). From the results of the analysis by Srivastava et al, it can be concluded that the applicability of any one algorithm to the other two depends to a large extent on their use. In the same test environment, the YOLO algorithm outperformed both SSD and Faster R-CNN, making it the best of the three algorithms.

We used the selected algorithm (YoloV5) to compare with FireNet a popular artificial intelligence fire detection project on GitHub. To achieve fairness in the experiments, both algorithms were trained using the same dataset and tested for performance using the same test set.

FireNet Model Summary: 261 layers, 61508200 parameters, 0 gradients, 154.6 GFLOPs. Model size is 117.9MB.

| Class | Images | Labels | P     | R     | mAP@.5 | mAP@.5:.95 |
|-------|--------|--------|-------|-------|--------|------------|
| All   | 1018   | 1273   | 0.709 | 0.648 | 0.658  | 0.319      |
| Fire  | 1018   | 710    | 0.564 | 0.52  | 0.485  | 0.212      |
| Smoke | 1018   | 563    | 0.704 | 0.581 | 0.632  | 0.298      |

Speed: 0.2ms pre-process, 33.5ms inference, 2.4ms NMS per image at shape (32, 3, 640, 640)

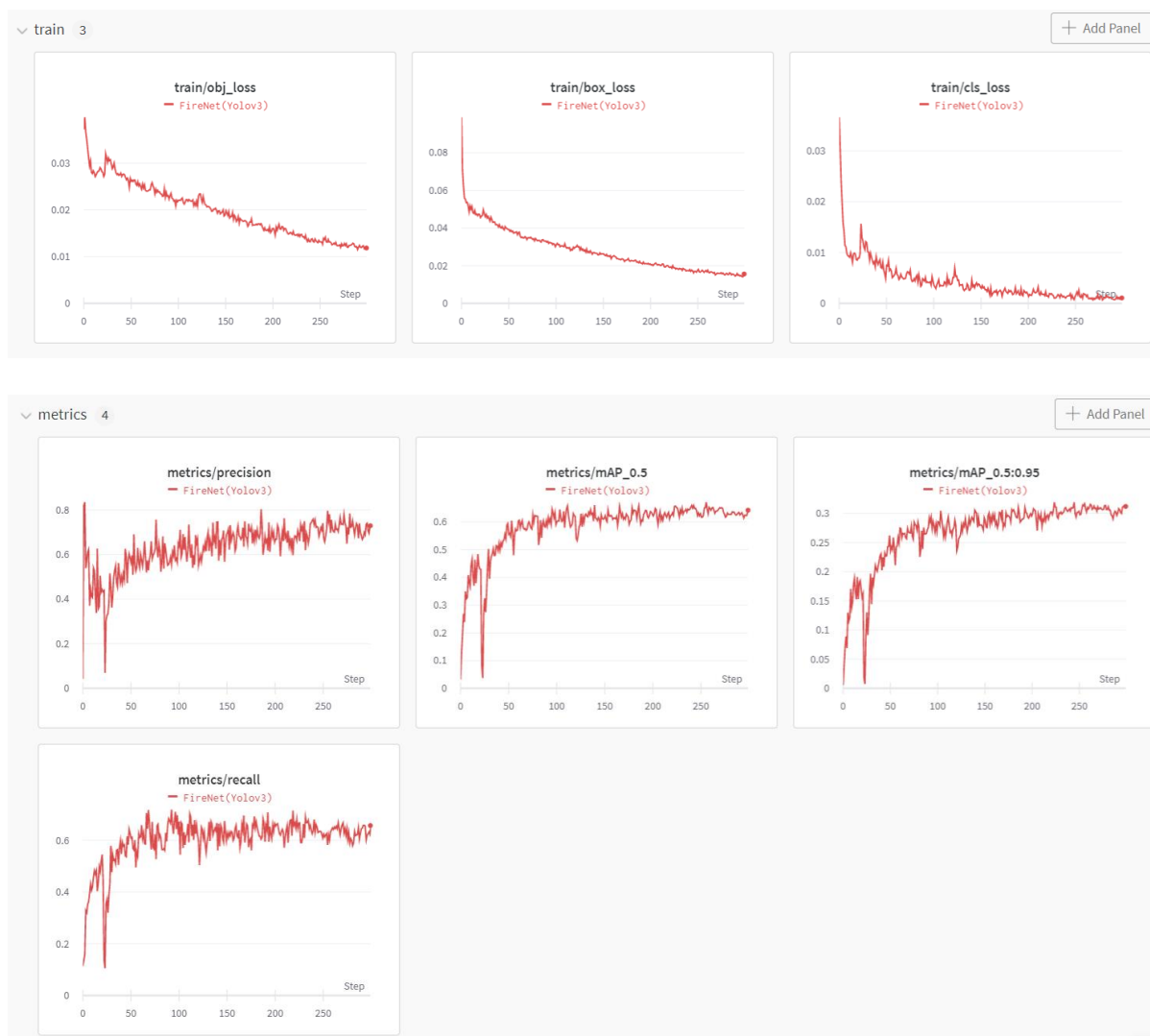


Figure 52: FireNet Model Summary Training Results

YoloV5 Model summary: 213 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs. Model size is 13.8MB.

| Class | Images | Labels | P     | R     | mAP@.5 | mAP@.5:.95 |
|-------|--------|--------|-------|-------|--------|------------|
| All   | 1018   | 1273   | 0.86  | 0.796 | 0.829  | 0.575      |
| Fire  | 1018   | 710    | 0.78  | 0.741 | 0.748  | 0.711      |
| Smoke | 1018   | 563    | 0.941 | 0.851 | 0.911  | 0.739      |

Speed: 0.5ms pre-process, 6.2ms inference, 1.3ms NMS per image at shape (32, 3, 640, 640)

In the above experimental data, we can see that the performance of the original YOLOv5 is significantly improved compared to FireNet, with the trained model being only 13.8 MB, which is more than 8 times smaller than the FireNet model. This will be much simpler when deployed on devices with lower computing power.

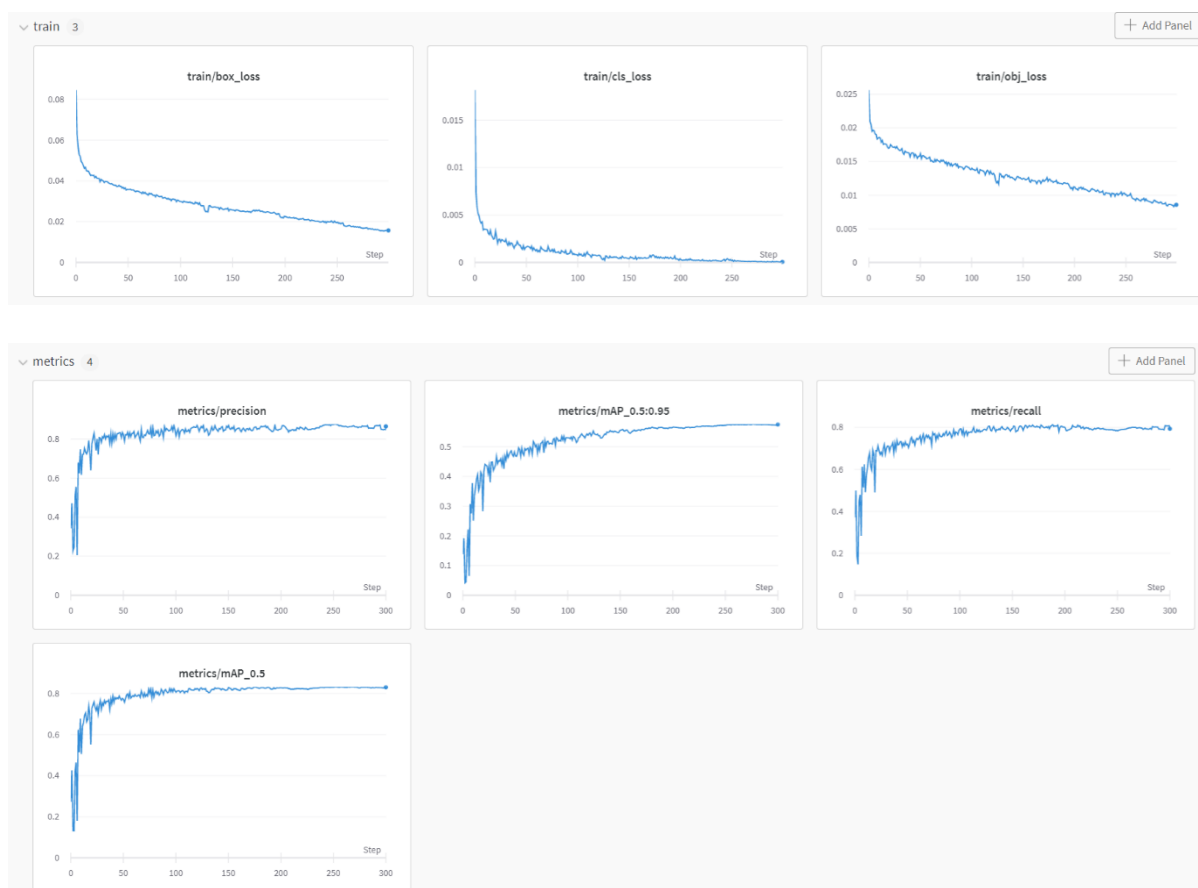


Figure 53: YOLOV5 Model Summary Training Results

The model at this point is deployed directly on the Raspberry Pi and can achieve a real-time video detection frame rate of about 2FPS using the CPU for computation.

This proposal conducts a series of ablation experiments based on YOLOv5 to make it lighter (smaller Flops, lower memory usage, fewer parameters), faster (adding shuffle channel, yolov5 head for channel cropping, and inference speed can reach 10+FPS on Raspberry Pi 4B at least at 320 input size), and easier to deploy (taking off Focus layer and 4 slice operations to let the model quantization accuracy drop within an acceptable range).

After completing the ablation experiments on the YOLOv5 model, the algorithm can perform real-time target detection in 1080p HD quality at over 7FPS on the Raspberry Pi 4B.





Figure 54: 1080p HD quality at over 7FPS on the Raspberry Pi 4B

Experiments were performed using candles in both bright and dark environments using the already modified model and Algorithm to test the smallest detectable pixel values.



Figure 55: The experiments result in the dark environments.

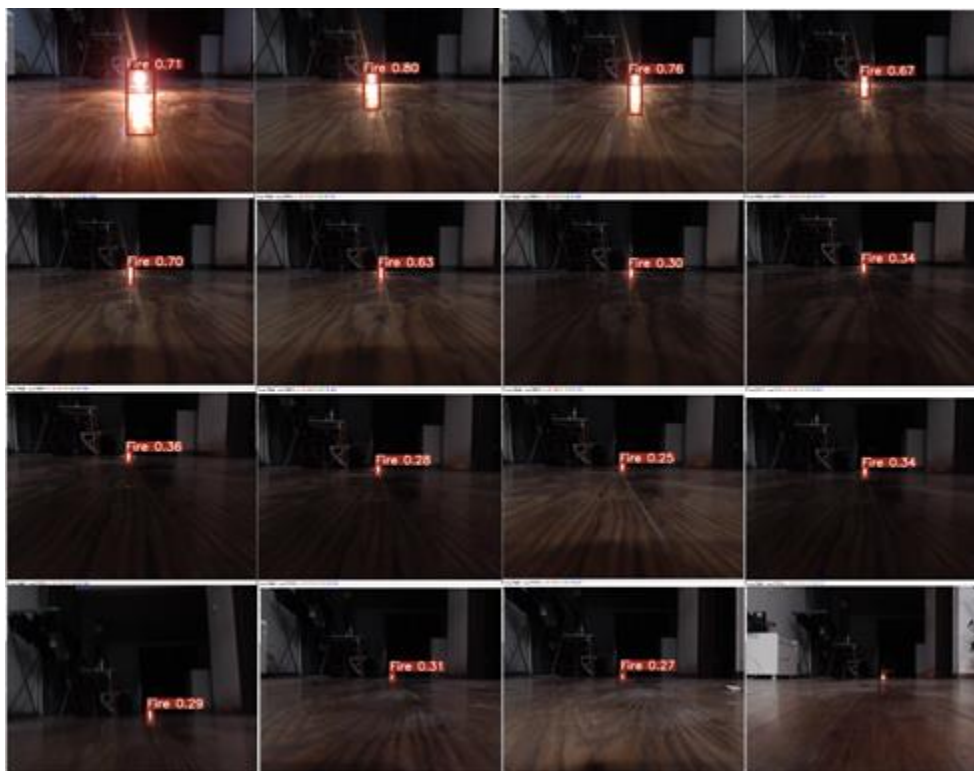


Figure 56: The experiments result in the bright environments.

| Distance/m | Confidence (1.00 for 100%) |            |
|------------|----------------------------|------------|
|            | Lights-on                  | Lights-off |
| 0.5        | 0.81                       | 0.71       |
| 1.0        | 0.54                       | 0.8        |
| 1.5        | 0.51                       | 0.76       |
| 2.5        | 0.47                       | 0.67       |
| 3.0        | 0.53                       | 0.70       |
| 3.5        | 0.46                       | 0.63       |
| 4.0        | 0.55                       | 0.3        |
| 4.5        | 0.40                       | 0.34       |
| 5.0        | 0.32                       | 0.36       |
| 5.5        | 0.31                       | 0.28       |
| 6.0        | 0.26                       | 0.25       |
| 6.5        | 0.28                       | 0.34       |
| 7.0        | 0.38                       | 0.29       |
| 7.5        | 0.48                       | 0.30       |
| 8.0        | 0.35                       | 0.31       |

|     |   |      |
|-----|---|------|
| 8.5 | / | 0.27 |
| 9.0 | / | 0.26 |
| 9.5 | / | /    |

When **light is off** the smallest detectable pixel value is 10.4x10.4

When **light is on** the smallest detectable pixel value is 15.6x10.4

The maximum distance it can detect the frame when the light is off is 9.0 m whereas when light is on the maximum distance is 8.0 m.

**For Greenhouse Applications the algorithm performs within the desired KPIs because it is difficult to find a greenhouse with heights higher than 8 m.**

## 2.3 Insect Detection

Following the same methodology as presented in section 2 and using the dataset as presented on Deliverable 4.1 then Yolo was trained for Insect Detection. Figure 57, presents the dataset for Insect Detection Yolo Model Training whereas Figure 58 presents the metrics over epochs.

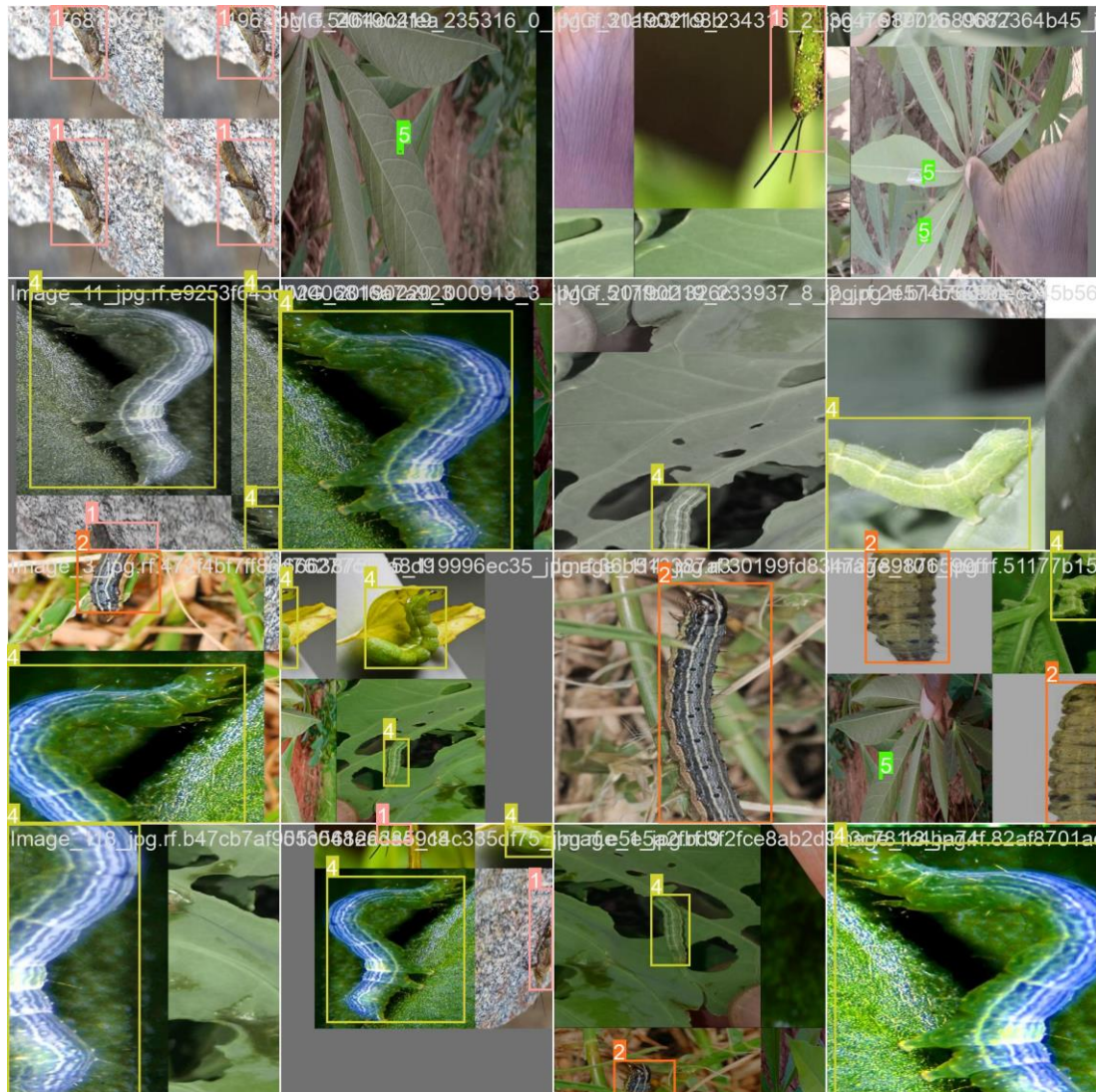


Figure 57: Dataset for Insect Detection Yolo Model Training

Epoch in machine learning is a time-based metric that defines the number of times a model has been run. It's especially useful for tracking model performance over time, and can be used to detect when a model is starting to plateau

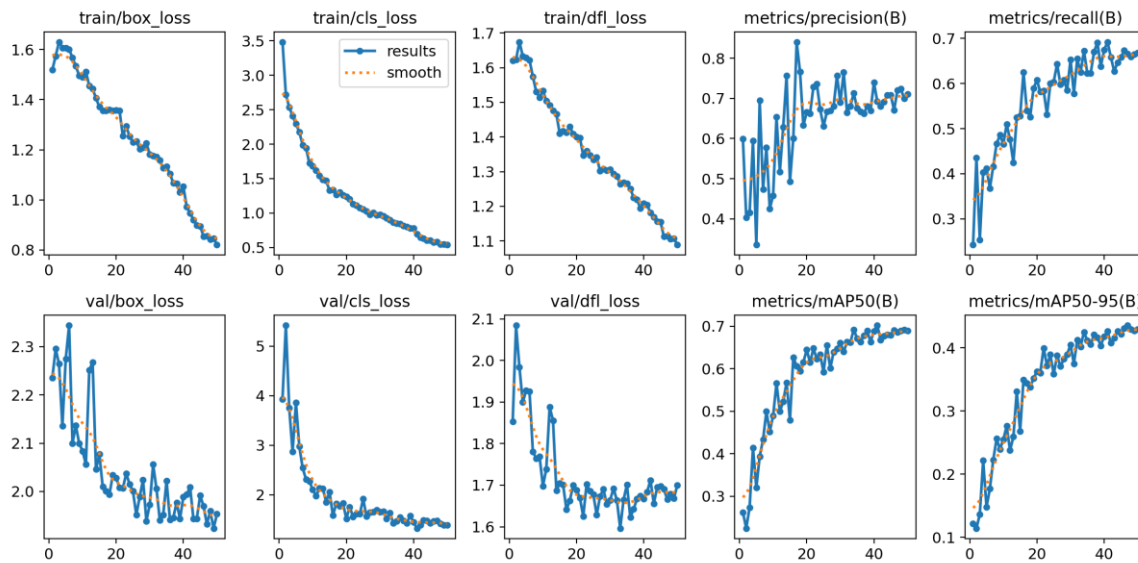


Figure 58: Metrics over epochs

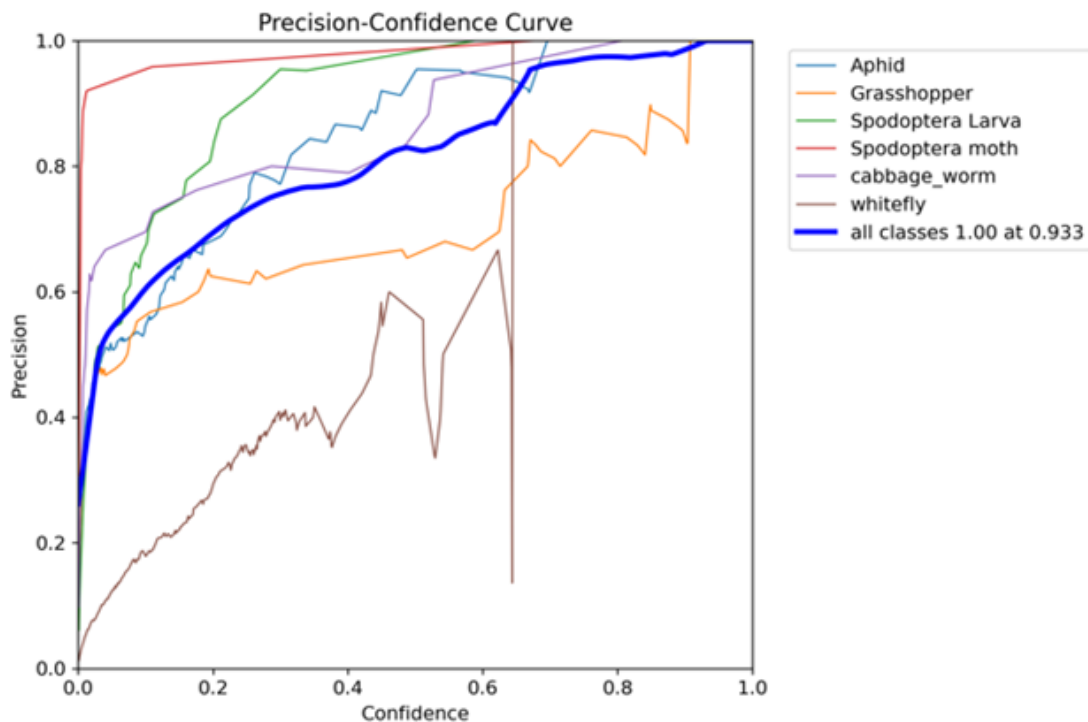


Figure 59: Precision Confidence Curve

We see that precision is bounded between 0 and 1. It increases when the threshold is increased. We also note that precision can be made arbitrarily good, as long as the threshold is made large enough. Hence precision alone cannot be utilized to assess the performance of a classifier. We need a second metric: the recall. Precision-Recall curves summarize



the trade-off between the true positive rate and the positive predictive value for a predictive model using different probability thresholds.

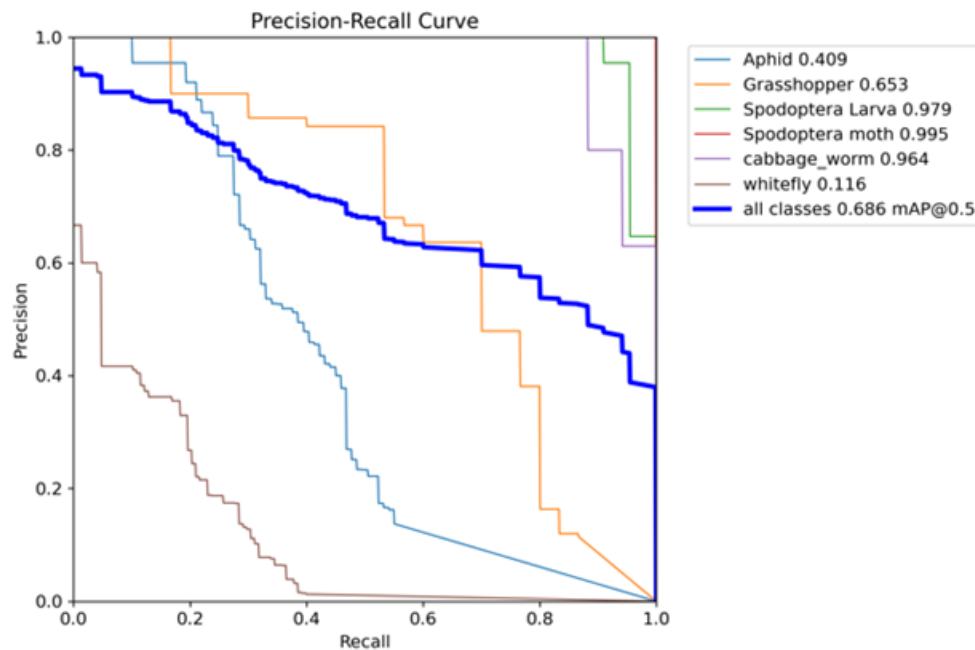


Figure 60: Precision - Recall Curve

As shown from Figure 60, the precision-recall curve is obtained by plotting the precision on the y-axis and the recall on the x-axis for all values of the threshold between 0 and 1. We have seen that for very high thresholds (high means a little smaller than 1.0) the precision was very high and the recall was very low. This is represented by the red dot in the upper left corner of the graph. For very low thresholds (a little larger than 0.0) we have shown, that the recall was almost 1.0 and the precision was identical to the ratio of positive samples in the dataset.

The results for insect detection are presented on Figures 61 to 63. The detected insects are enclosed in a bounding box.

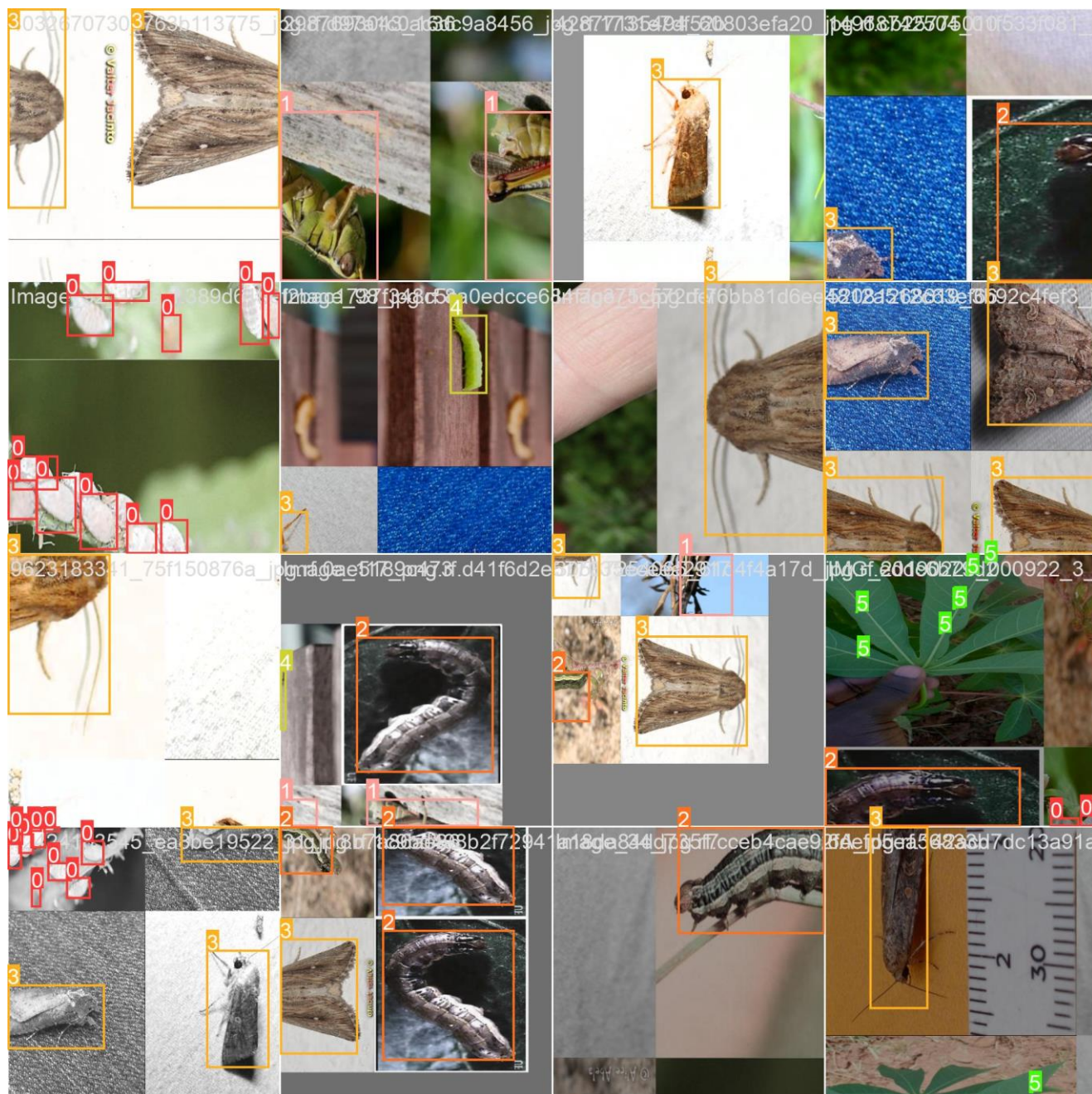


Figure 61: Insect Detection Results



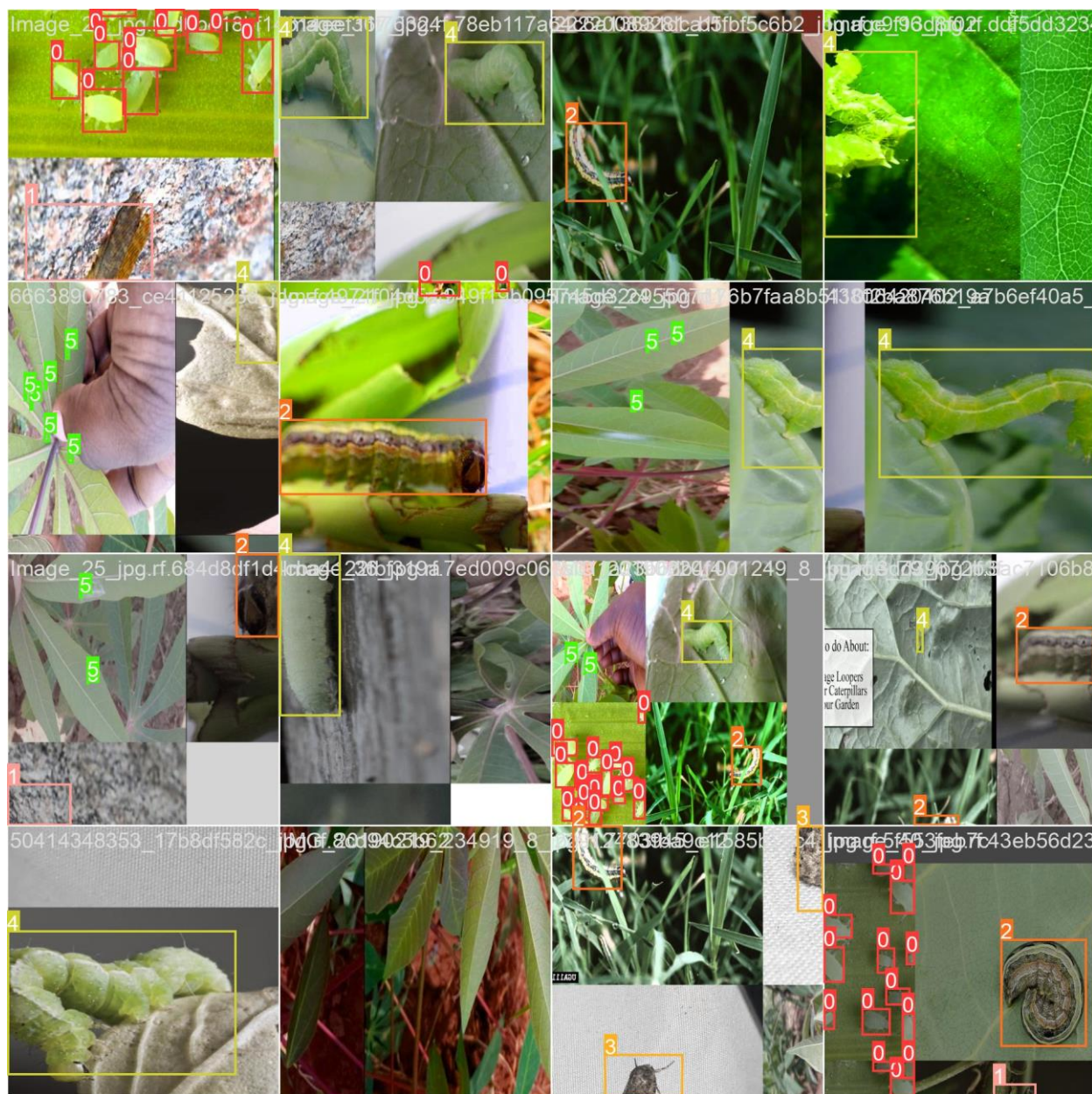


Figure 62: Insect Detection Results 2



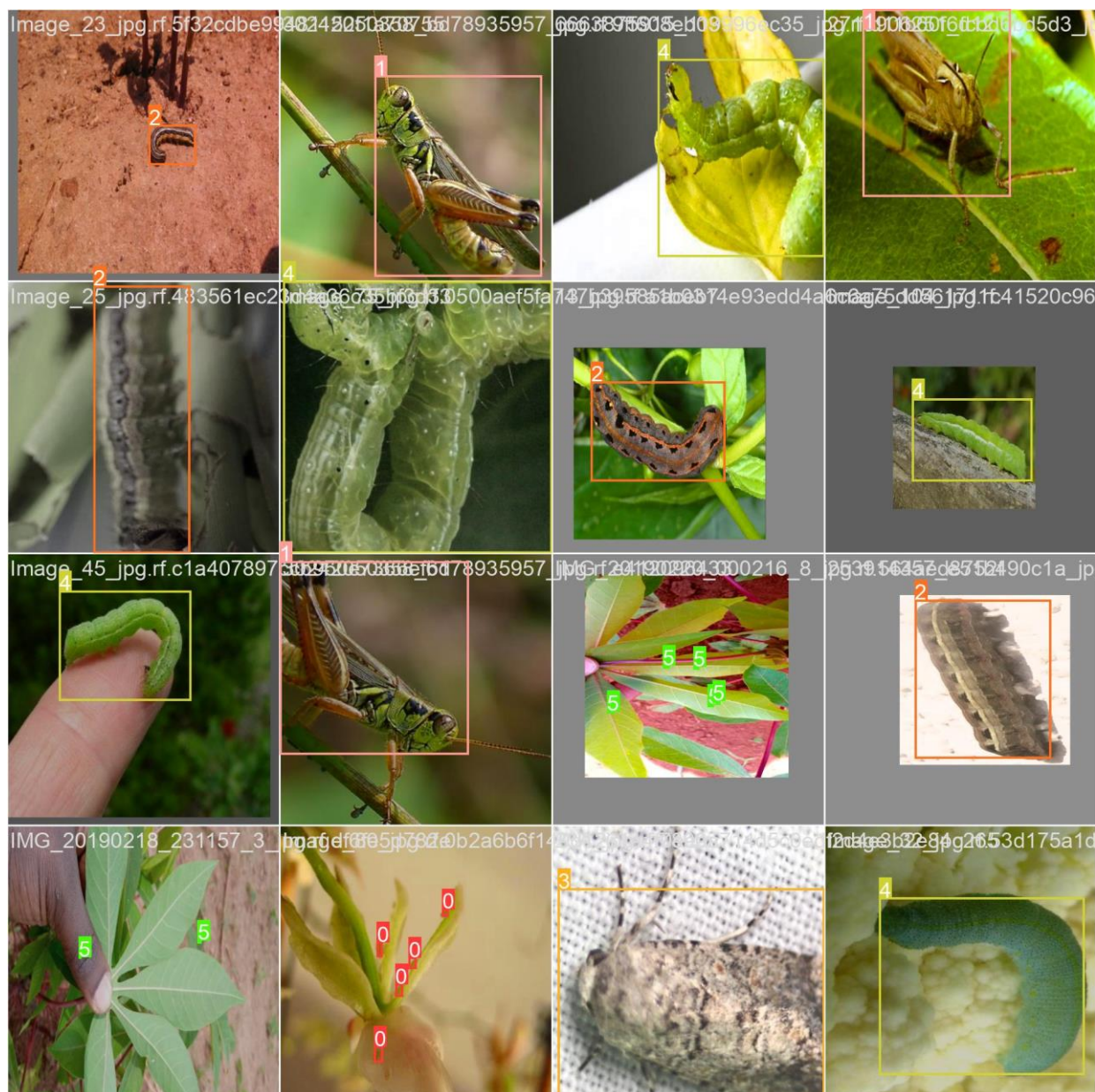


Figure 63: Insect detection Results 3

The training of YOLO algorithm and the Detection of Insects has been successful and meets the set KPI requirements.

### 3 Multispectral Imaging

The KPIs for the multispectral imaging required a sensitivity analysis and tolerance to interferences due to UAV motor vibrations as well as the presence of wind and wind gusts. Furthermore, spectral data should be analysed using any relevant software.

#### 3.1 Sensitivity Analysis

For this part of the sensitivity analysis it was decided to run the experiment outdoors for the following reasons:

1. Outdoors it is possible to run the UAV at full speed to test the worst case scenario of interference due to vibrations. Whereas, in a greenhouse it is not expected to use full throttle.
2. Outdoors the presence of wind and wind gusts is possible whereas in a greenhouse it is not.

The experiments were on November 20<sup>th</sup>, 2023, between the hours of 3:30 pm to 5 pm, at a field across the street from the university. Prior to the flight the location was verified that it was not included in any flight restricted zone as shown on Figure 64.

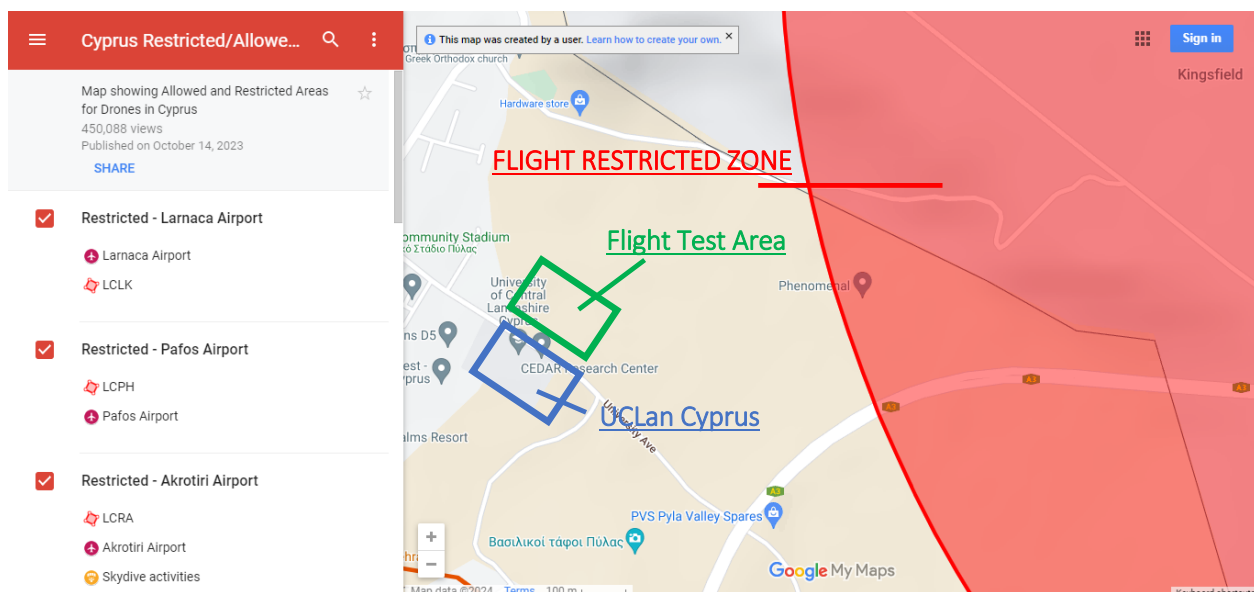


Figure 64: Flight test Area Verification for any Restrictions

As can be seen from Figure 64, the flight test area is indicated by colour green, is across the street from the university UCLan Cyprus which is indicated by colour blue and it is away from the Flight Restricted zone. Also, worth noting that our test flight was much lower than the allowable 120 m. The test flight altitude was in the range of 30-50 m.

Furthermore, the flight was performed by a licenced UAV pilot as shown on figure 65.





**ΠΙΣΤΟΠΟΙΗΤΙΚΟ ΟΛΟΚΛΗΡΩΣΗΣ ΗΛΕΚΤΡΟΝΙΚΗΣ ΕΚΠΑΙΔΕΥΣΗΣ**  
**PROOF OF COMPLETION OF THE ONLINE TRAINING**

Όνομα (First name)  
**IGOR**  
Αριθμός αναγνώρισης (Identification number)  
**CYP-RP-373w237ziwx2**

Επώνυμο (Last name)  
**LYSENKO**  
Ισχύει μέχρι (Expiration date)  
**08/11/2028**



Figure 65: Licence Certificate of UAV Test Pilot

The flight plan as presented on Figure 66 was created on the DJI flight controller and was aimed at scanning the indicated area. The importance of the selected area was its composition which included both parts full of vegetation and other dry parts without vegetation.

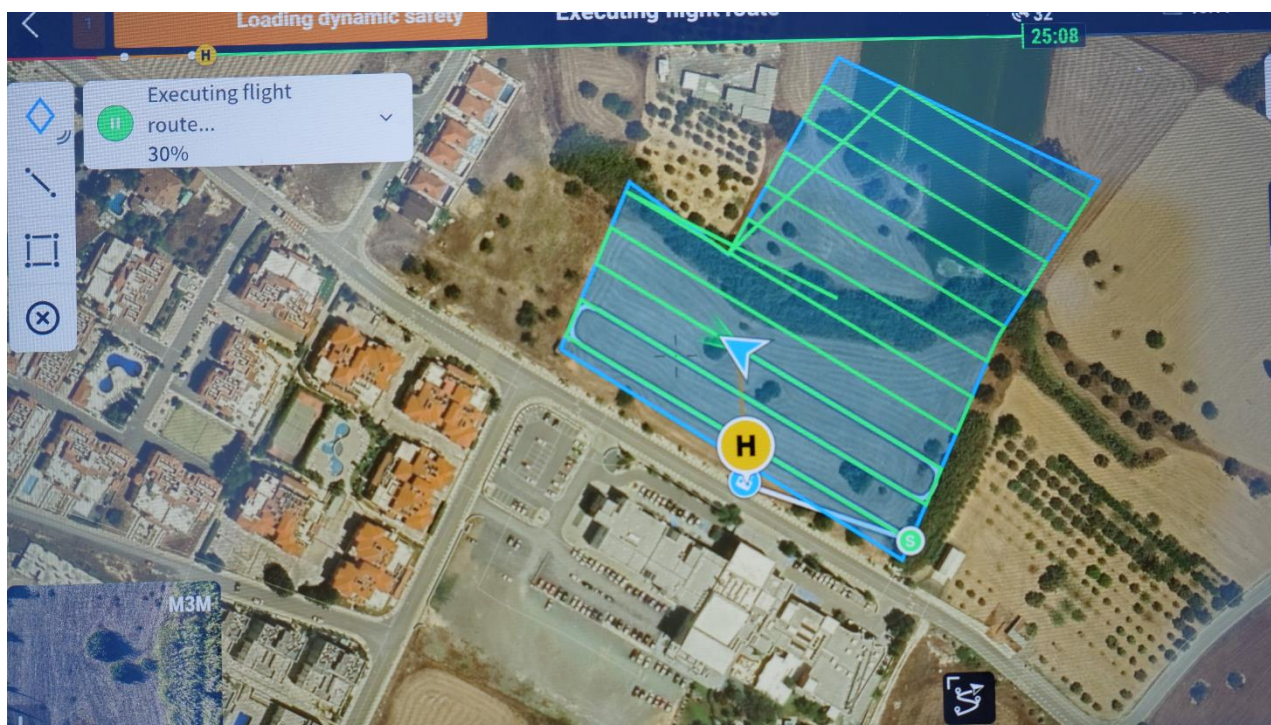


Figure 66: Flight Plan

The UAV take-off and landing are presented by Figures 67 and 68 respectively. Worth noting that Figure 40 also shows the compact size of the UAV.



Figure 67: UAV Take-Off



Figure 68: UAV Landing

The flight duration was approximately 35 minutes and it was constantly monitored on the flight controller with live telemetry statistics of battery status as well as any notifications and warnings. The notifications and warning usually include the presence of strong winds greater than the UAV's capabilities. At no instance during the entire flight duration we received any warnings.

#### November 20, 2023

|               | High    | Low     | Average |
|---------------|---------|---------|---------|
| Temperature   | 68.4 °F | 51.1 °F | 63.0 °F |
| Dew Point     | 50.0 °F | 42.4 °F | 46.3 °F |
| Humidity      | 77 %    | 41 %    | 55 %    |
| Precipitation | 0.01 in | --      | --      |

|                | High     | Low      | Average  |
|----------------|----------|----------|----------|
| Wind Speed     | 24.0 mph | 2.0 mph  | 11.6 mph |
| Wind Gust      | 35.0 mph | --       | 17.0 mph |
| Wind Direction | --       | --       | NW       |
| Pressure       | 29.92 in | 29.73 in | --       |

Figure 69: Weather Conditions the Day of Test Flight

The weather conditions and forecast for the day of the test flight from Figure 69. The weather conditions were obtained online from accuweather, (<https://www.accuweather.com/en/cy/pyla/123792/november-weather/123792?year=2023>). As shown an average temperature of 63 °F (17.2 °C) and an average humidity in the range of 55%. The temperature was within the UAV operating conditions (-10 to 40 °C) hence it was considered as ideal conditions for a flight. On the other hand, the day of the test flight was windy. The average wind speed was 11.6 mph (5.2 m/s) and the average wind gust was 17 mph (7.6 m/s). Both values were within the UAV specifications for safe handling and smooth right. Even the maximum wind speed of 24 mph (10.7 m/s) was within the UAV range (upto 12 m/s = 26.8 mph). The presence of wind can also be clearly shown on Figure 70 with the swaying of the trees.





Figure 70: Presence of Wind – Swaying Trees

November 20, 2023

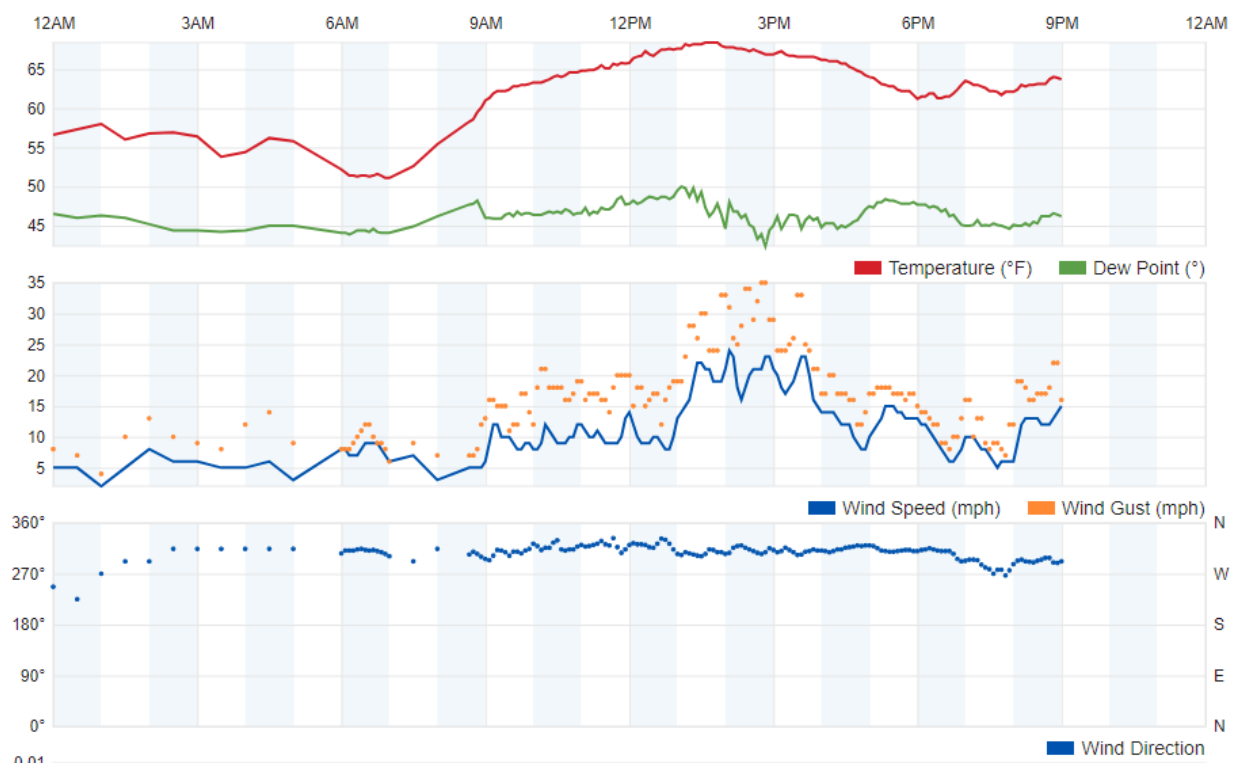


Figure 71: Hourly Weather Forecast

Further, investigation of the peak wind gust forecast, revealed that the peak wind gusts could possibly reach 35 mph which was not within the UAVs capabilities. Hence, the hourly forecast was investigated for the test flight feasibility. As shown on Figure 71, between the hours of 13 to 15:30 pm the possible wind gusts reaching peaks of 35 mph were a prohibiting factor to proceed with the test flight. However after 15:30 pm both the wind and wind gusts were forecasted to decrease considerably and be within the UAV capability range. Hence, as soon as we observed that the forecast was correct and between the hours of 13:30 to 16 pm the wind gusts considerably reduced then we proceeded with the test flight. The wind direction remain constant North-West at 270°.

The flight was successful and the DJI Mavic 3M proved its airworthiness. Furthermore, the DJI proved that the gimbal system provides stability and immunity from vibrations due to wind and UAV motors. **Successful flight, stability and immunity from vibrations were conditions set by KPIs.**

## 3.2 Multispectral Image Analysis

### 3.2.1 DJI Terra Software

The multispectral data from DJI Mavic 3 Multispectral was analysed using the DJI Terra Software. The software mapping outputs are presented on Figures 72 to 77. The vegetation index maps include NDVI, GNDVI, NDRE, LCI and OSAVI. These maps can be produced to provide insight into plant growth and health.

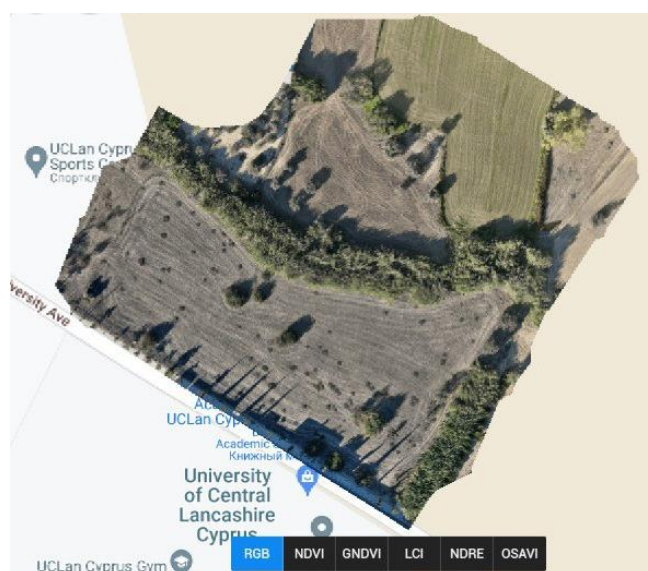


Figure 72: Terra Mapping Output – RGB

Figure 72 shows the visual (RGB) mapping of the scanned area. The mapping is constructed by adding hundreds of images taken during the test flight. The bird-eye view clearly shows that there are areas of dense vegetation and other areas of thinner vegetation and no-vegetation.

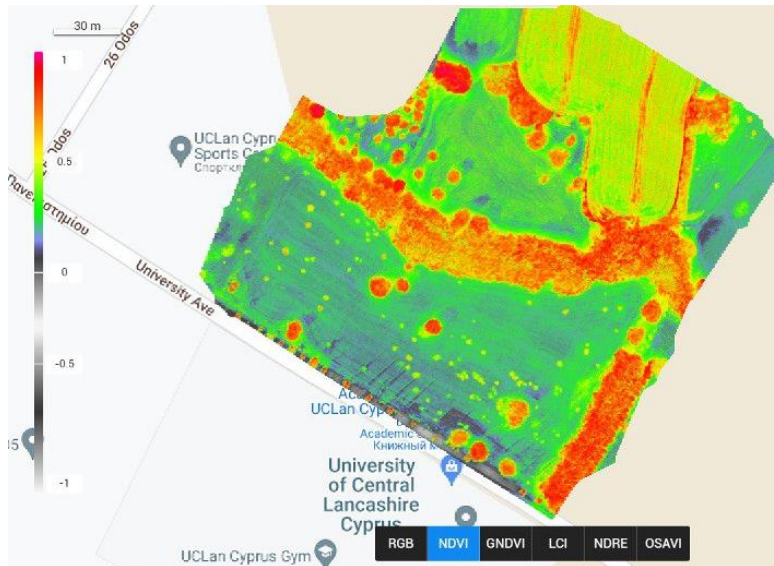


Figure 73: Terra Mapping Output – NDVI

NDVI index from 0.6 – 0.9 is related to dense vegetation similar to tropical forests or crops at the growth peak. This is shown with colour red. And agrees with the visual results of Figure 72.



Figure 74: Terra Mapping Output – GNDVI

GNDVI can serve as an indicator of photosynthetic activity within the vegetation cover and identify variation and levels of greenness in the crop. GNDVI is also more effective in estimating nitrogen and moisture content in the crop canopy.



GNDVI is used in evaluating mature vegetation. The results of Figure 74 with red colour identifies mature vegetation which is also verified by the visual results of Figure 72.



Figure 75: Terra Mapping Output – LCI

This LCI index shown on Figure 75 is used to calculate the amount of chlorophyll in the leaves and is more reliable in the late summer months since the patterns are correlated with the crops' final yield.



Figure 76: Terra Mapping Output – NDRE

The selection of NDRE of Figure 76 might be preferable between the middle and late growth stages when the leaves of the crop have a denser concentration of chlorophyll and the REDGE wavelength will be more absorbed than the red wavelength. However, it may not be as effective as NDVI for the evaluation of larger land areas.



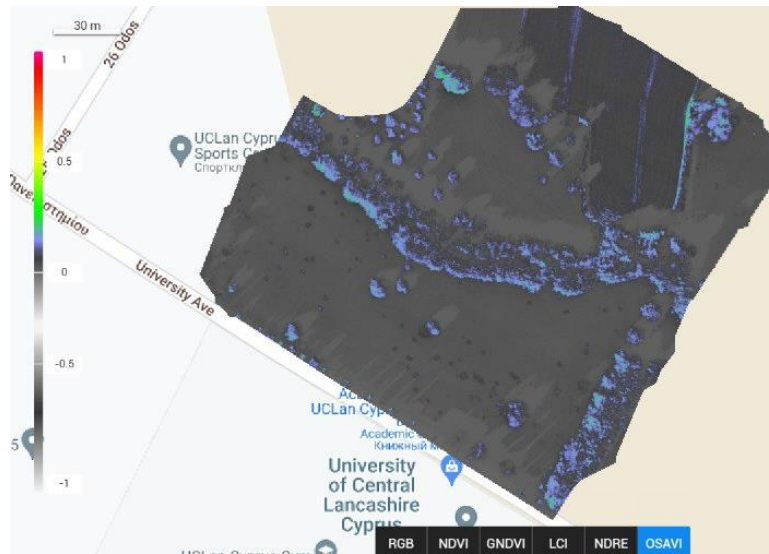


Figure 77: Terra Mapping Output – OSAVI

OSAVI index shown on Figure 77 is found to be more effective in areas with sparse vegetation where there is visible soil. With black colour the soil area is indicated whereas heavy vegetation is in blue. This data agrees with results of figure 72.

### 3.2.2 Open Source Software QGIS

QGIS, also known as Quantum GIS, is a geographic information system (GIS) software that is free and open-source. QGIS supports viewing, editing, printing, and analysis of geospatial data.



Figure 78 Visual Image 0119

The QGIS software offers histograms with the index results.

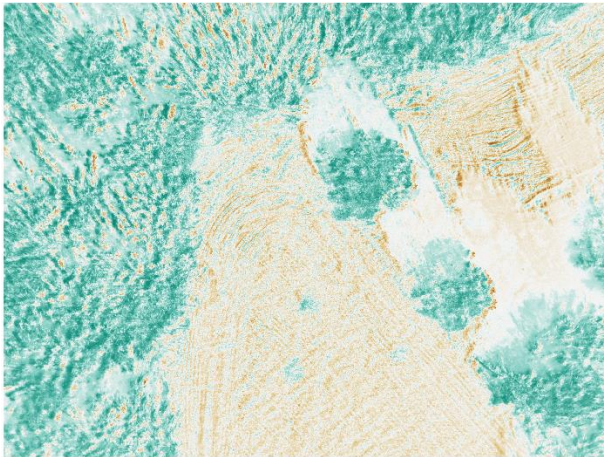


Figure 79 NDVI for 0119

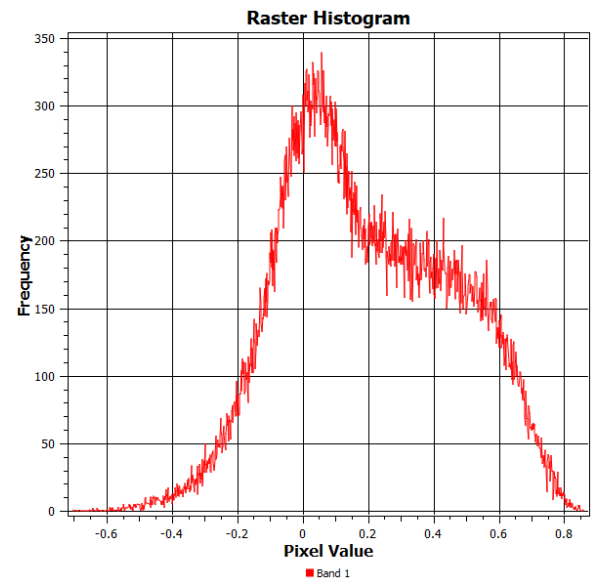


Figure 80 NDVI histogram for 0119

The histogram of Figure 80 shows that the majority of the pixels shown on Figure 79 are filled with positive values indicating mostly green vegetation.

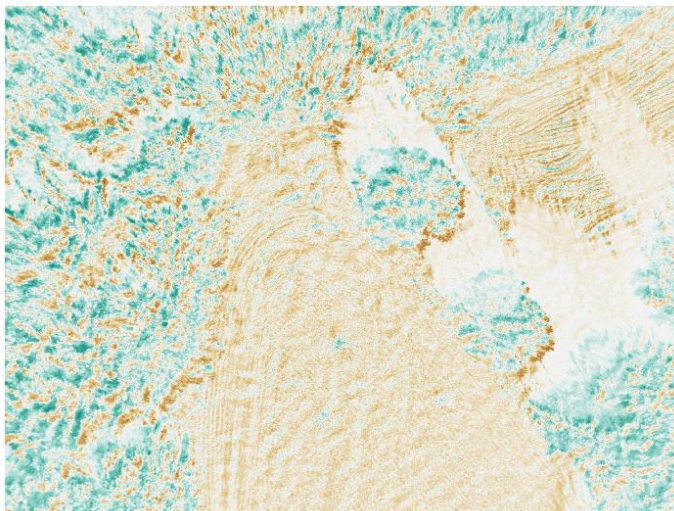


Figure 81 NDRE for 0119

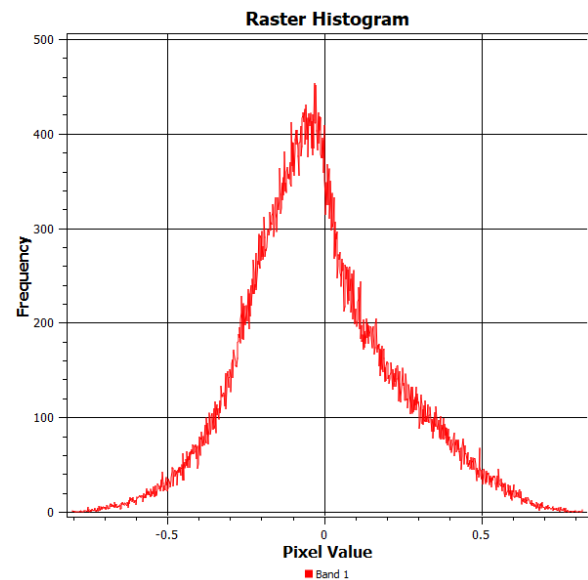


Figure 82 NDRE Histogram for 0119



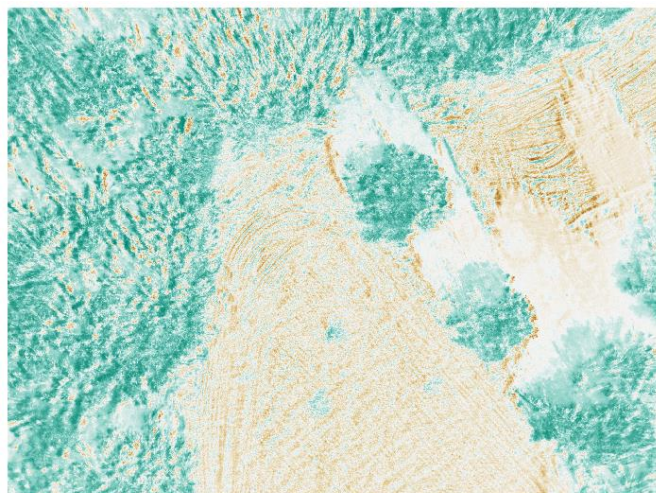


Figure 83 OSAVI for 0119

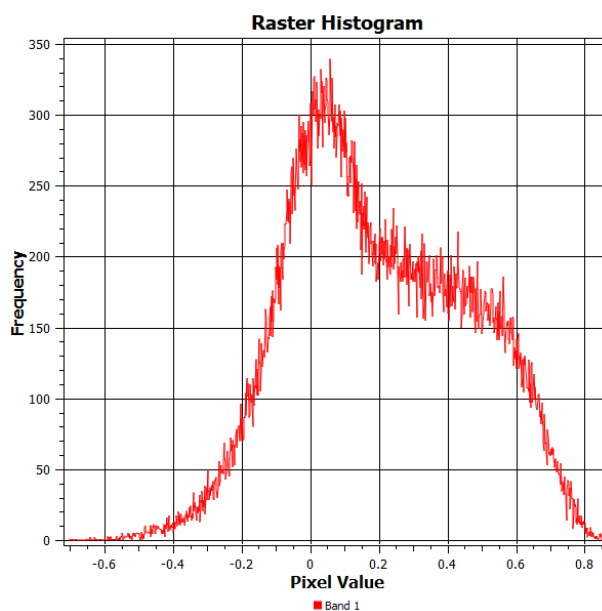


Figure 84 OSAVI Histogram for 0119

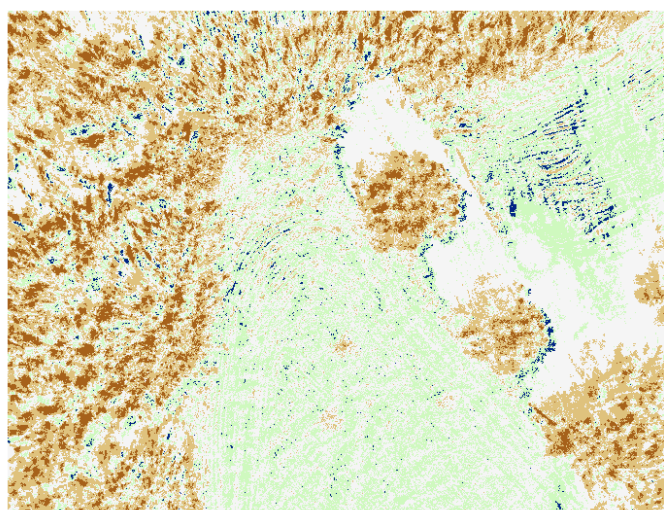


Figure 85 NDWI for 0119

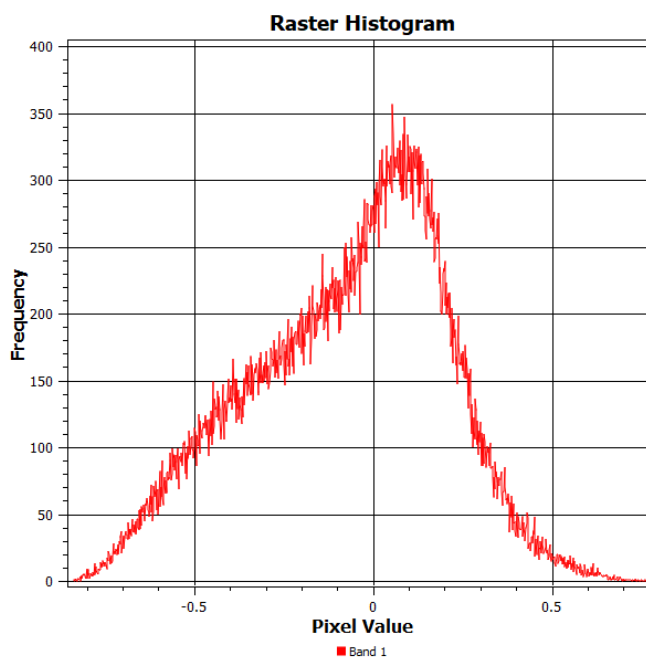


Figure 86 NDWI histogram for 0119

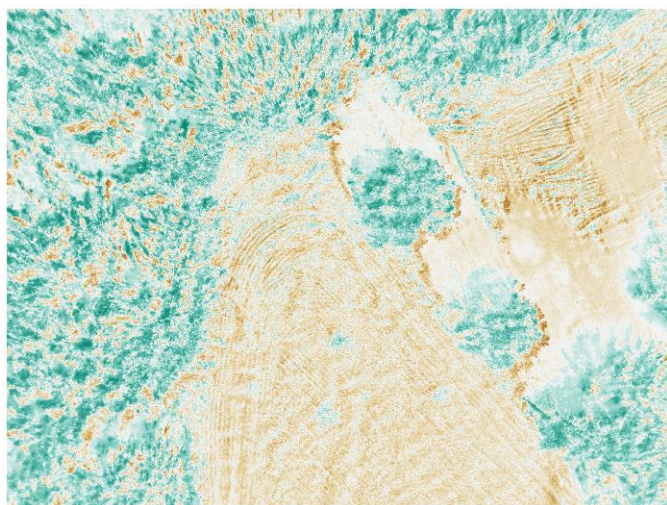


Figure 87 GNDVI for 0119

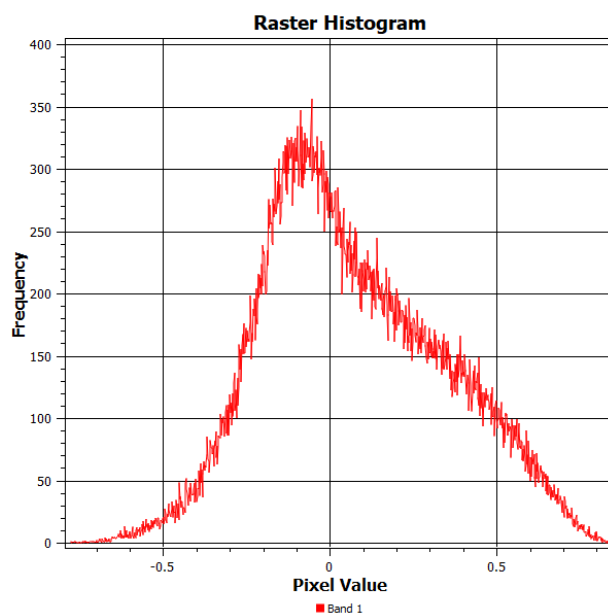


Figure 88 GNDVI histogram for 0119

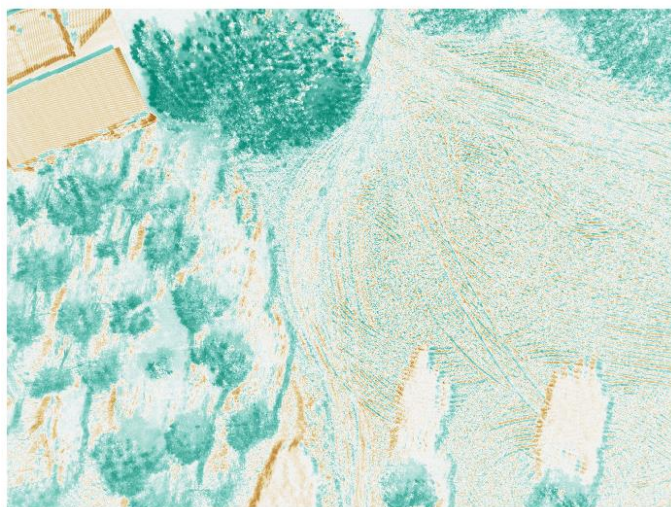


Figure 89 NDVI for 0136

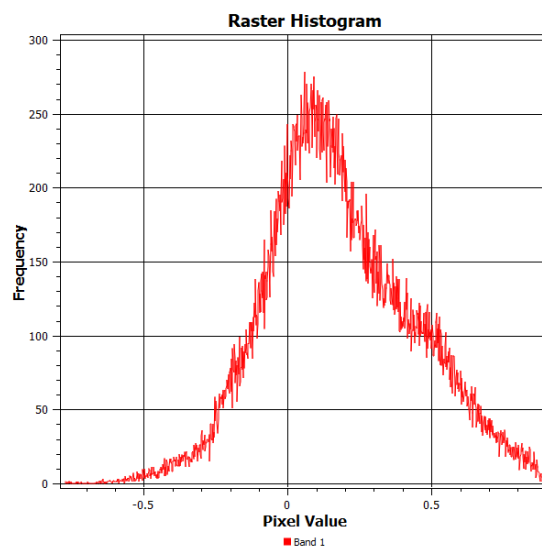


Figure 90 NDVI histogram for 0136



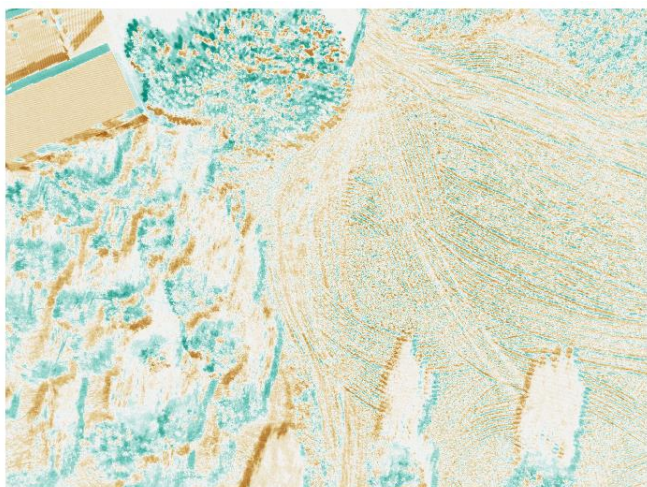


Figure 91 NDRE for 0136

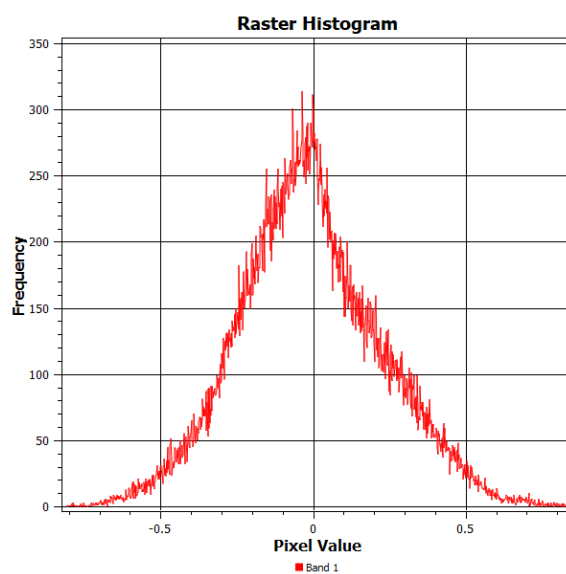


Figure 92 NDRE for 0136

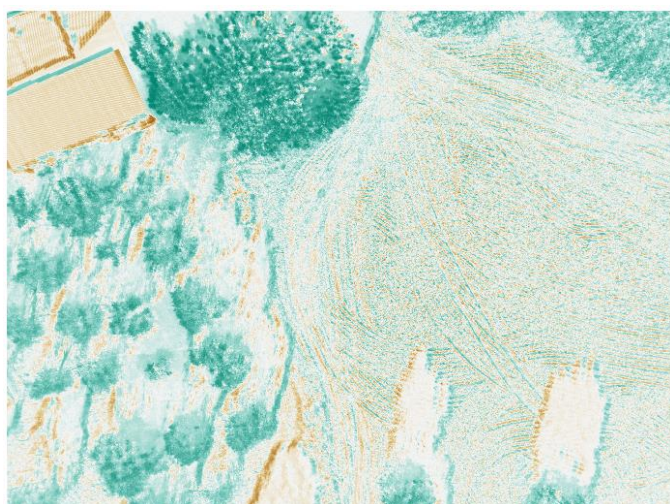


Figure 93 OSAVI for 0136

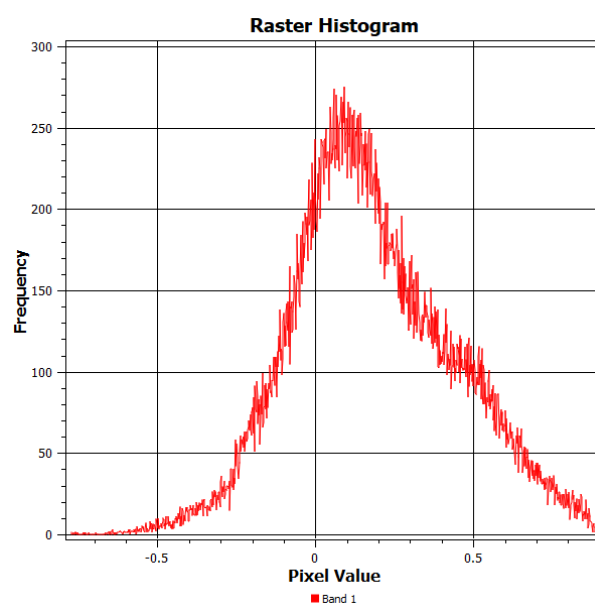


Figure 94 OSAVI histogram for 0136

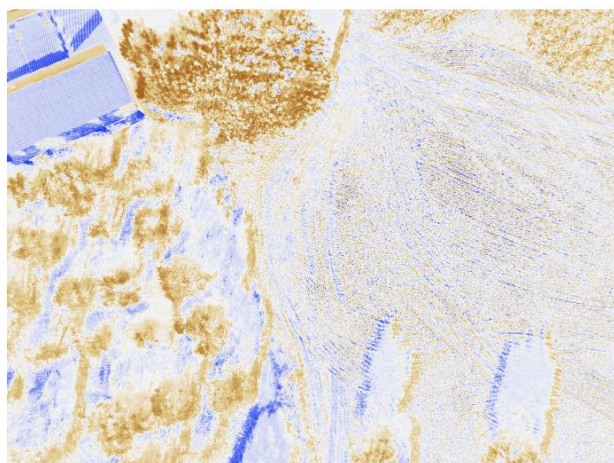


Figure 95 NDWI for 0136

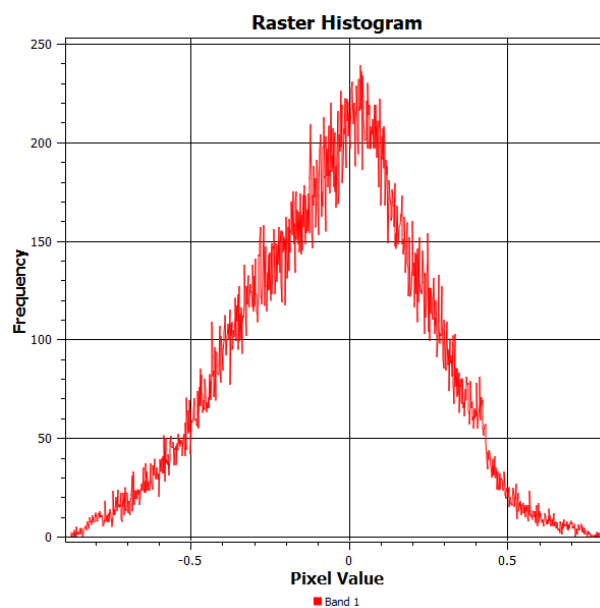


Figure 96 Histogram for 0136



Figure 97 GNDVI for 0136

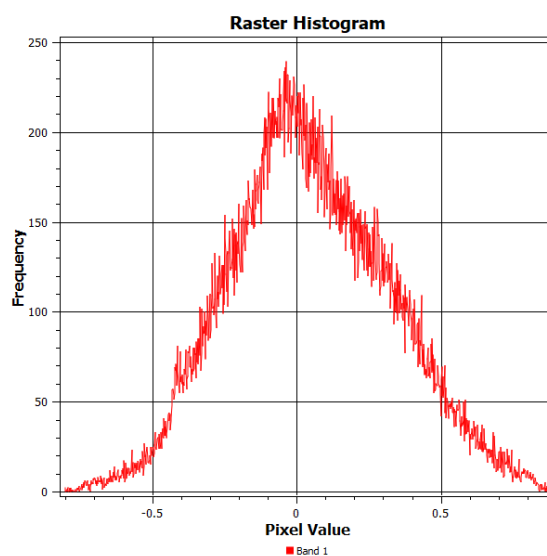


Figure 98 Histogram for 0136

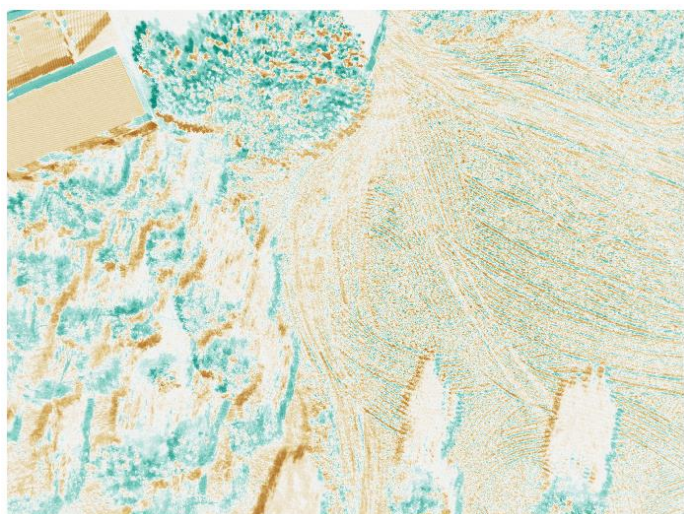


Figure 99 LCI for 0136

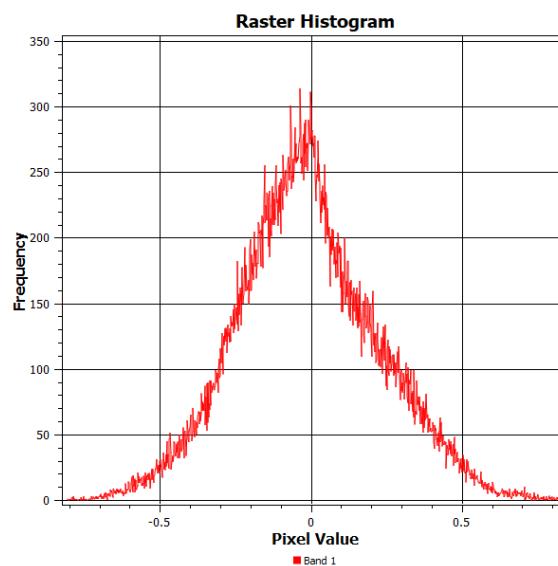


Figure 100 LCI histogram for 0136

The UAV flight and collection of data using a multispectral camera, along with the data analysis using the software DJI Terra and open source QGIS has been successful and meets the set KPI requirement.



## 4 Indoor Positioning

Indoor positioning was tested using AI cameras and mmWave Radars. The advantage of the AI camera is that the type of object UAV or UGV can also be detected and also calculate its indoor location. On the other, mmWave radar hardware can only detect objects without distinguishing what it is. However, a network of mmWave equipment can increase the accuracy considerably.

### 4.1 3D Indoor Positioning using AI Cameras

Following the methodology of Section 2, the YOLO algorithm was trained to detect and identify UGVs and UAVs. However, the optimised YOLO algorithm proposed in section 2, could not be used. The optimised version which was running on single board computer, Raspberry Pi 4 could not work with the algorithm that was utilising the stereo capability of the OAK-D camera. This was not a problem, because the optimised version was necessary to provide high performance for small UAV applications where weight, volume and power are strict constraints. On the other hand, the application of 3D indoor positioning aims at providing stationary cameras which can detect a UAV or UGV and calculate its location within a room.

#### 4.1.1 Experimental Setup

Two cameras were placed in a laboratory. The laboratory contained benches and chairs. One camera was placed close to the floor whereas the second one was placed on the ceiling. Ground truth marks were placed on the floor. The experimental setup is visualised on figure 101.

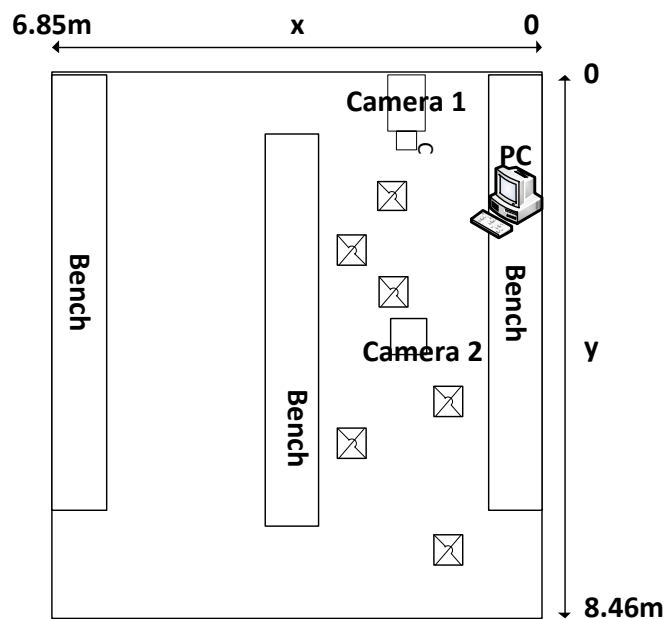


Figure 101: Experimental Setup



Table 5: Location of Cameras and Ground Truth Markers

|              | X     | Y     | Z    |
|--------------|-------|-------|------|
|              | (cm)  | (cm)  | (cm) |
| <b>CAM 1</b> | 198   | 14    | 11   |
| <b>CAM 2</b> | 197   | 398   | 348  |
| <b>P 9</b>   | 157   | 733   | 0    |
| <b>P 10</b>  | 256   | 617   | 0    |
| <b>P 11</b>  | 145.5 | 483   | 0    |
| <b>P 12</b>  | 187   | 387   | 0    |
| <b>P 13</b>  | 239.5 | 283.5 | 0    |
| <b>P 14</b>  | 207   | 125   | 0    |
| <b>P 15</b>  | 148.5 | 261   | 0    |

Table 5 tabulate the coordinates (x, y, z) of the cameras and the ground truth markers whereas Figure 3 visualizes then in 3D space. All the ground truth markers are on floor (z=0) whereas camera 2 is installed at the ceiling at a height of 348 cm.

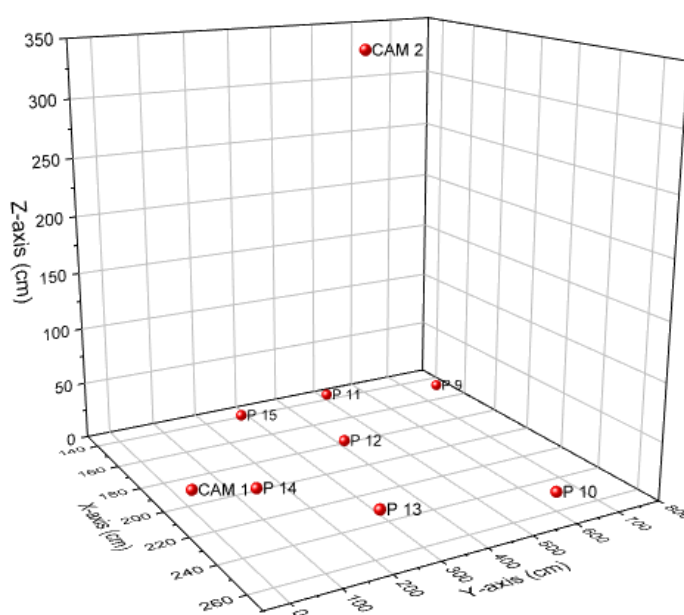


Figure 102: Cameras and Ground Truth Markers in 3D Space

#### 4.1.2 Distance acquisition with stereo cameras

For this project, the YOLO algorithm was deployed to the OAK stereo camera to determine the distance between the detected fire and the camera. The OAK (OpenCV AI Kit) camera's depth calculation is based on stereo vision. (Luxonis, 2022) Two identical monochrome global shutter cameras are positioned side by side in the OAK camera, capturing two slightly different views of the same scene. Using a process known as stereo matching, these perspectives are then utilised to derive depth information.

Stereo matching involves finding corresponding pixels in the two images that represent the same point in the real world. The difference in x-coordinates between these pixels is then utilised to compute the depth of the related point. The wider the gap, the closer the point is to the camera.

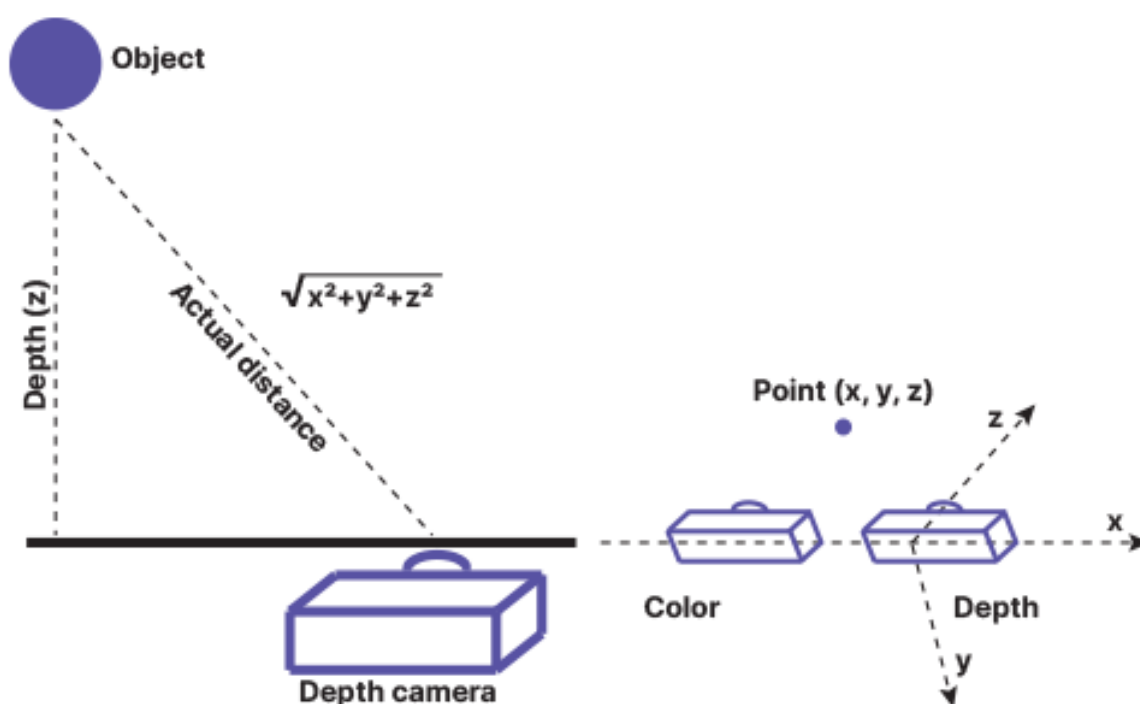


Figure 103: Because the depth map contains the Z distance, objects parallel to the camera are measured exactly as standard. For objects that are not parallel, the Euclidean distance can be calculated using the Euclidean distance. (Luxonis, 2022)

The difference is the difference between the horizontal locations of the equivalent pixels in the OAK camera's left and right pictures. It is a critical input parameter in stereo matching and is used to derive scene depth information.

To determine the difference between the corresponding pixels in the left and right pictures, the OAK camera employs a semi-global matching (SGM) method. The SGM method generates a greyscale image known as a difference map, in which each pixel indicates the difference between the corresponding pixels in the left and right images. The difference value may be used to generate a depth map, which depicts the metric distance of each pixel from the camera.

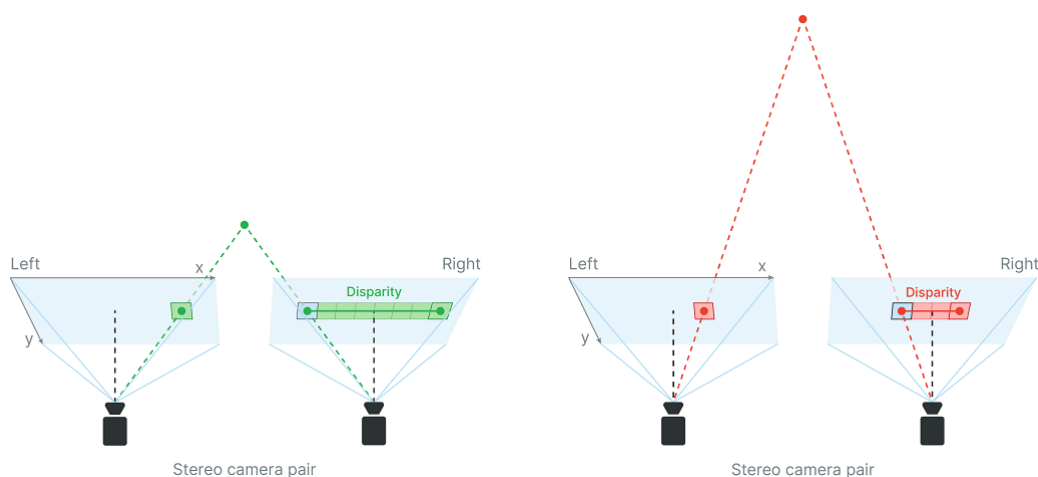


Figure 104: The disparity refers to the distance between the two corresponding points in the left and right images of a stereo pair. (Luxonis, 2022)

Using the x, y and z information obtained by the camera, the horizontal ( $x\_degree$ ) and vertical ( $y\_degree$ ) angles between the target and the camera can be calculated by geometric methods. Figures 105 and 106 shows a simulated fire detection using the proposed system.



Figure 105: Obtaining the distance, angle offset and other information using the proposed system.

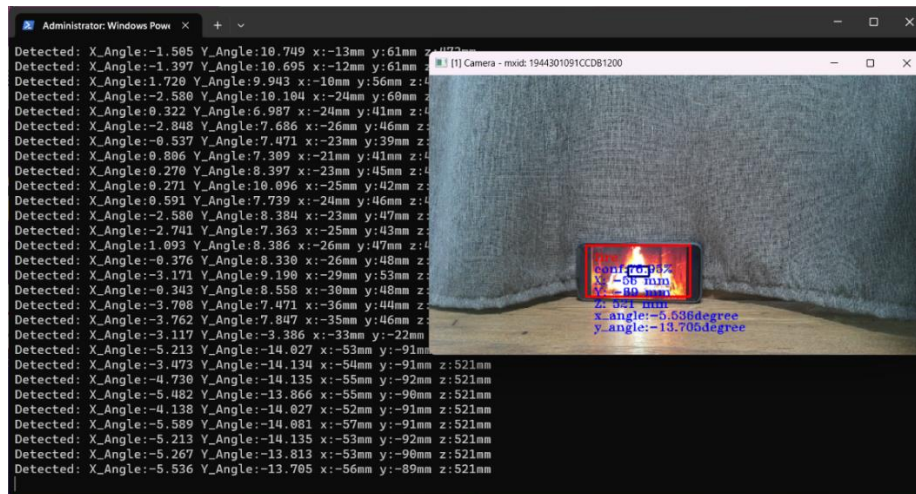


Figure 106: Outputting the detected information to the terminal.

### 4.1.3 Stereos Depth Mapping

Two algorithms in OpenCV can achieve: **STEREOBM** & **STEREOSGBM**. Both algorithms rely on the input images to be rectified, which is taken the results of the calibration phase applied them and made sure that images line up nicely.

**STEREOBM** is the fastest of the two, which use black matching. It works by taking a block of pixels in the left hand frame scans across the corresponding x-axis in the right-hand frame then find the closest matching block between the two frames, the result of this is a disparity value per block which roughly is the distance in pixels between matching blocks.

**STEREOSGBM** using semi-global block matching, for each block of pixels in the left-hand frame, can scan across multiple directions in the right-hand frame to find the closest match. This approach is a lot more intensive but tends to produce a much more accurate depth map.

However, both algorithms don't give exactly the depth map straight away, to get the depth distance, the distance between two cameras to calculate the distance.

### 4.1.4 Stereo camera calibration

Calibrate stereo camera, because some cameras can produce distortions in the images they capture such as causing straight lines to appear curved, which will introduces error when doing computer vision.

This distortion can be computed and corrected by taking a load of photos with a chess board pattern in the frame, then pass those photos into its calibration API that gives back a matrix defining the camera's distortion which then can be used to correct for it.

#### 4.1.5 Conversion from Camera to Local Coordinate System

The camera coordinate system (x, y, z) requires to be converted to the local system as shown by figure 107.

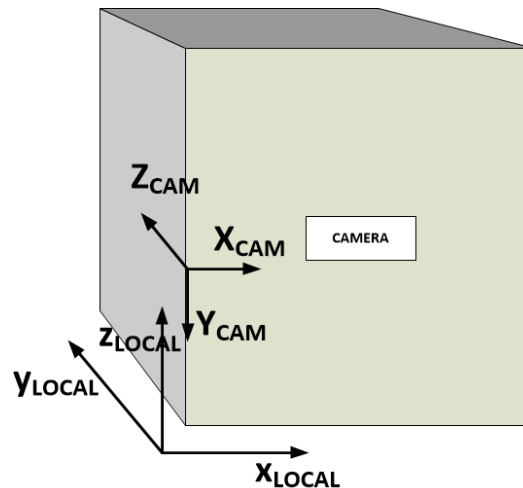


Figure 107: Camera and Local Coordinate Systems

The conversion is as follows:

$$Z_{CAM} = Y_{LOCAL}$$

$$X_{CAM} = X_{LOCAL}$$

$$Y_{CAM} = -Z_{LOCAL}$$

#### 4.1.6 Experimental Results

For the experiments we used 3 UGVs, one UAV and a basketball. Screenshots from the results are shown on Figures 108 to .....

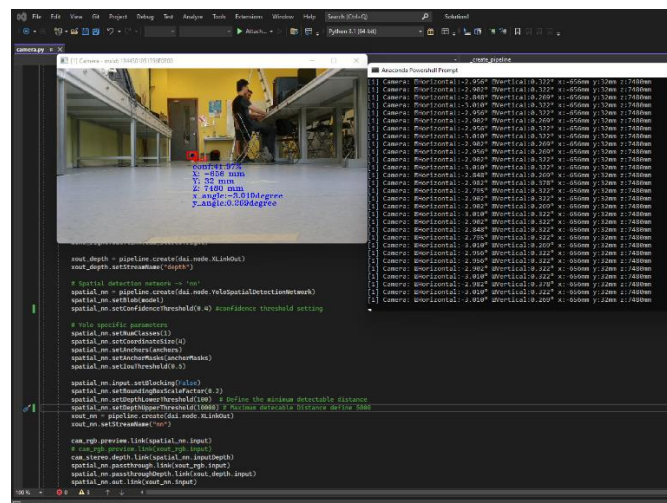


Figure 108: Example of Experimental Procedure



A screenshot of the experimental process is captured by Figure 108. As can be seen when the python code is executed then two more windows open on the screen. One window shows the camera view and the other the measurements. The measurements do not start until an object is detected first and then the location is calculated. When an object is detected then a bounding box is enclosing the detected object. The calculated values are then displayed at a rate of 30Hz. The YOLO network was trained to detect UGVs, UAVs and basketballs. Unfortunately though, after the time consuming process of training the algorithm, it was realised that we have put the same labels for UAVs and UGVs. Unfortunately the label was GV for both. Figure 109 shows the birdeye view camera detecting the UAV with 92% confidence and displaying the local 3D indoor location. The accuracy is calculated at a later stage.

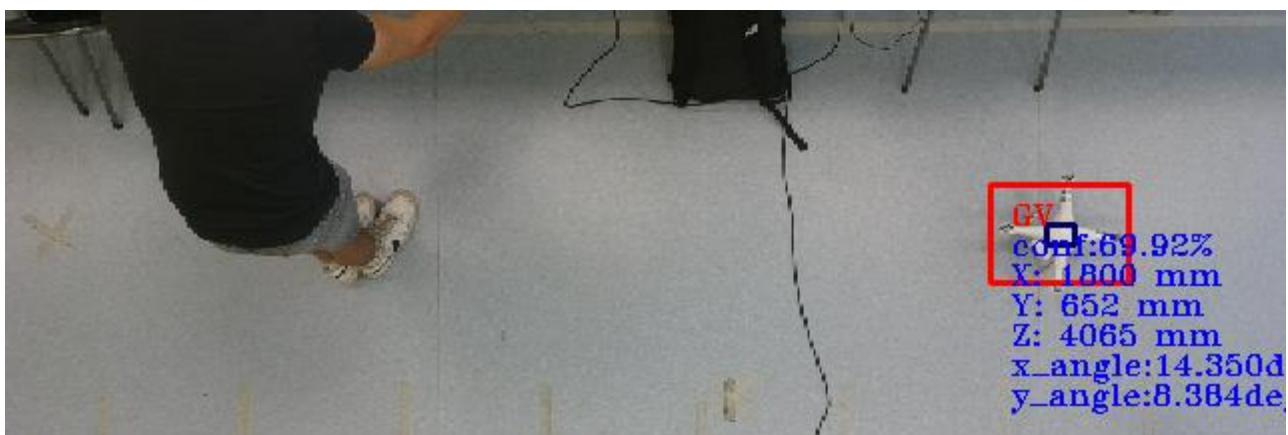


Figure 109: UAV Detected by Birdeye View Camera and Location is calculated

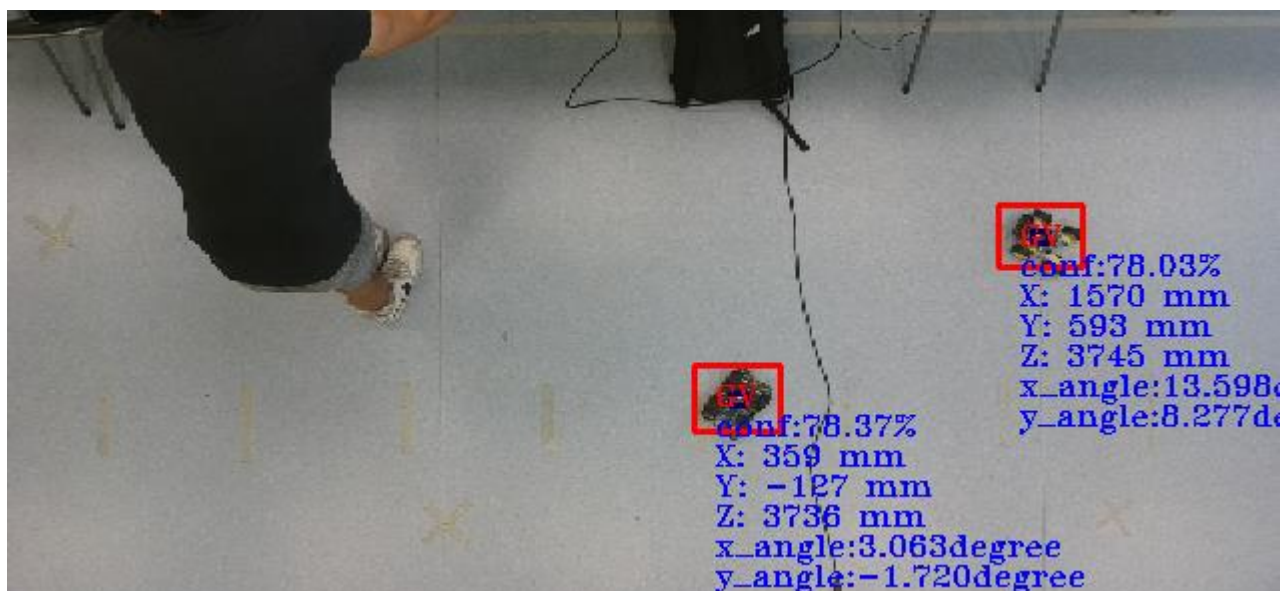


Figure 110: Detection of 2 UGVs Simultaneously



Figure 110 shows the detection of 2 UGVs simultaneously with confidence higher than 78%. Worth noting that the laboratory where the experiments were carried out was cluttered with equipment, desks, chairs and us walking around. This did not have a significant effect on the accuracy.

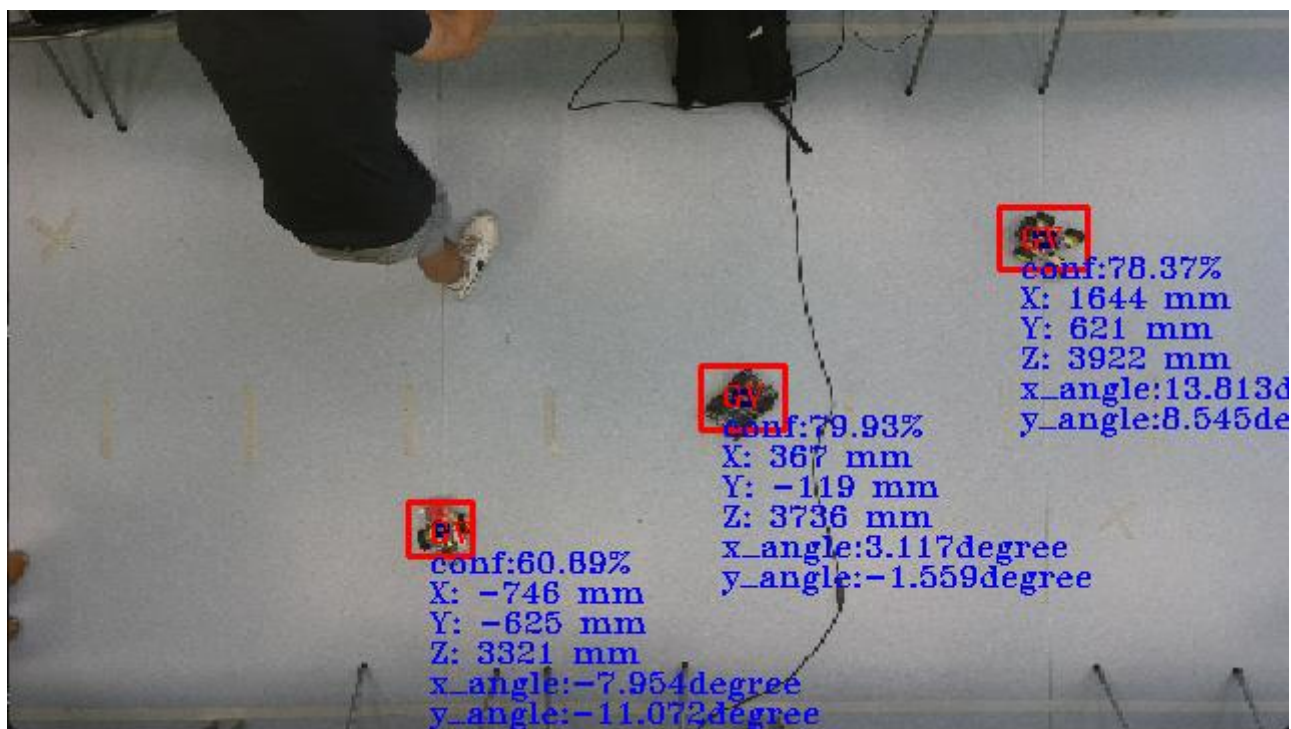


Figure 111: Detection of 3 UGVs Simultaneously

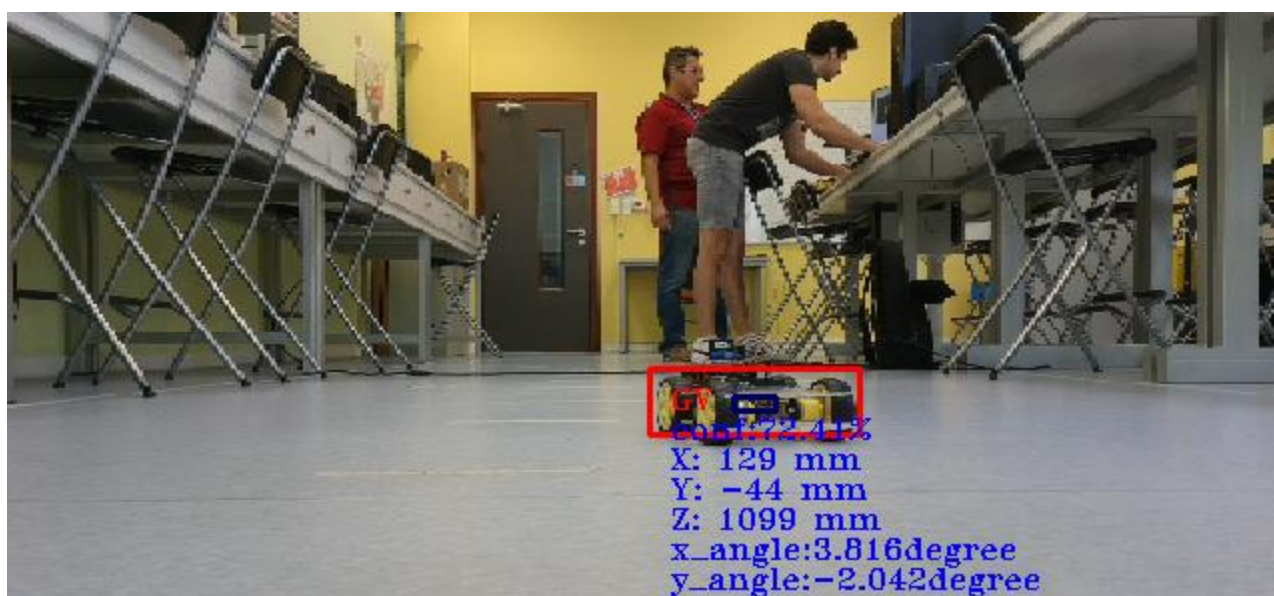


Figure 112: Profile Camera Detecting a UGV

Figure 112 shows a UGV detected by the profile camera. Worth noting that even though the UGV is on the ground the calculated height is proportional to the centre of bounding box around the object.



Figure 113: UAV Placed on a box to Verify Altitude Calculations

To verify the altitude calculations we place the UAV on a carton box. As shown from Figure 113, the altitude has increases compare to the results of figure 112.

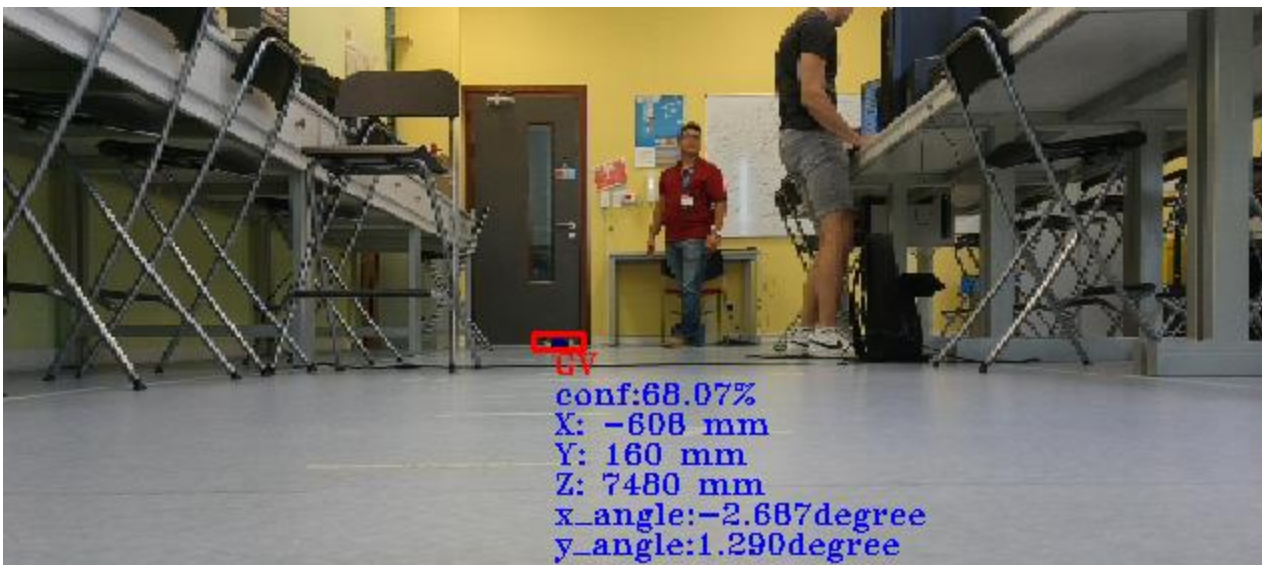


Figure 114: UGV Detection at a distance of 7.5m





Figure 115: Profile Camera Detecting 2 UGV Simultaneously

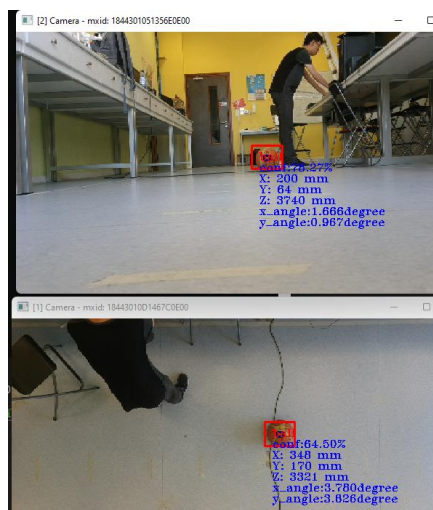


Figure 116: Both Cameras Simultaneously Detect the Basketball

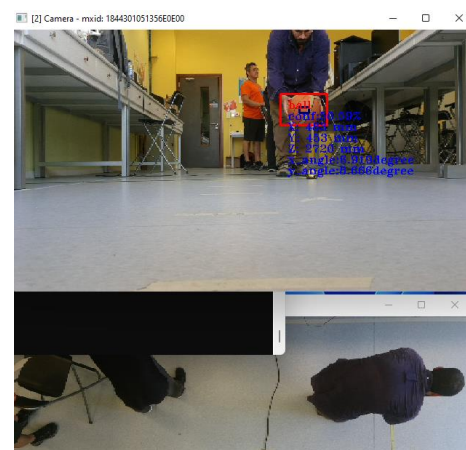


Figure 117: Basketball is hidden from the view of Birdeye Camera

Figure 116 shows both cameras being able to detect the basketball and calculate its location. On the other hand Figure 117 shows that the profile camera detects the basketball even though a person was holding it whereas the birdeye view camera cannot detect it since the person is blocking the view.

#### 4.1.7 Analysis of Experimental Data and Accuracy Calculations

Table 1 represents the error in cm for the analysis from the experimental data. As can be seen the maximum 3D error is 35 cm whereas the average is 22 cm and minimum is 5 cm.

Table 6: Localization Error

| Data Points | Error  |        |       |       |
|-------------|--------|--------|-------|-------|
|             | x      | y      | z     | 3D    |
|             | (cm)   | (cm)   | (cm)  | (cm)  |
| 1           | 19.60  | -9.00  | 15.70 | 26.68 |
| 2           | 18.10  | -15.00 | 25.60 | 34.76 |
| 3           | -22.00 | -24.60 | 10.20 | 34.54 |
| 4           | 2.90   | 1.10   | 3.56  | 4.72  |
| 5           | -5.60  | 11.50  | 1.30  | 12.86 |
| 6           | -3.50  | 16.00  | -4.70 | 17.04 |
| 7           | -2.10  | 18.00  | 6.60  | 19.29 |
| 8           | 10.30  | 13.60  | -6.18 | 18.15 |
| 9           | 2.40   | 23.80  | -6.18 | 24.71 |
| 10          | -19.20 | -16.20 | -1.24 | 25.15 |

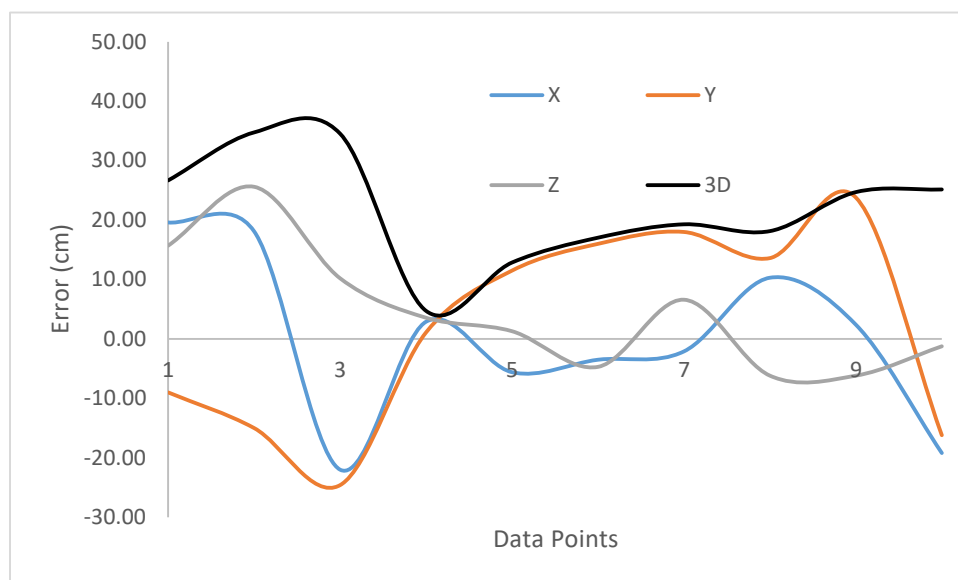


Figure 118: Localization Error

Figure 118, visualises the error in all directions. It can be seen that the error has deviations in all directions.

For indoor greenhouse application an average 3D error of 22 cm can be considered acceptable. Also even the maximum error of 35 cm satisfied the set KPI which was aiming at a sub-meter accuracy. Hence, the KPI requirements have been successfully met.

## 4.2 3D Positioning Indoor using mmWave Radar Sensors

The explosive growth of the Internet of Things (IoT) and the emergence of many Location-Based Services (LBS) and mobile smart applications make localization an even more important key-enabling technology in the Information and Communications Technology (ICT) world while many of these LBSs impose very high 3D localization accuracy requirements. Several approaches have been proposed during the last few decades to address the challenges of indoor localization however most of them only estimate positions on a horizontal ( $x$ - $y$ ) plane and many times neglect the vertical ( $z$ ) dimension. This lack of vertical information could lead into problems, such as the inability to determine whether a device is held up high or in a pocket etc., while accurate 3D positioning is also critical in scenarios such as drone-assisted crop seeding, search and rescue operations, and wireless communication (Han C. &, 2012), where sub-meter or cm-level accuracy is likely essential.

To address this demand, millimeter-wave (mmWave) positioning systems have emerged as a promising technology, offering high accuracy and robustness. mmWave is currently used in some Wi-Fi systems (e.g. IEEE802.11ad) while it is planned to be used in 5G communications due to its flexibility to use wider bandwidths and hence its strong potential in achieving much higher data rates and capacity. mmWave systems typically operate in frequencies between 26 to 100GHz. At those very high frequencies there is large availability of bandwidth which could lead to fine timing resolution and hence high ranging accuracy. The very small wavelength also allows the development of small and compact massive phase antenna arrays that enable the accurate estimation of angles (azimuth and elevation) of arrival. All this accurate context could be used for achieving cm-level 3D positioning accuracy or better (D. Wang, 2019). In this work, we capitalize on the potential of mmWave technology to accurately provide ranging and angling information, and sustain the momentum of ongoing research efforts in this topic by demonstrating its suitability to achieve cm-level accuracy, while presenting the most important challenges it imposes.

In response to the increasing demand for precise 3D indoor positioning in smart applications, there has been a growing surge in research and development efforts in recent years. These efforts are aimed at exploring advanced technologies to meet this need. The authors of (A. Sesyuk S. I., A survey of 3d indoor localization systems and technologies, 2022) offer a comprehensive survey of 3D indoor localization techniques and approaches. It delves into various modern technologies, providing insights and evaluations. Notably, the authors of this paper reference some relevant works. For example, in (A. Shahmansoori, 2018), the authors theoretically derive the Cramér-Rao Bound (CRB) for position and rotation angle estimation uncertainty using mmWave signals from a single transmitter, even in the presence of scatterers. They demonstrate that under open Line of Sight (LoS) conditions, it is feasible to estimate a target's position and orientation angle by leveraging information from multipath signals. However, this approach comes with a noticeable performance penalty. Additionally, the authors of (Y. Han, 2016) showcase the advantages of array antennas in determining a device's orientation. Notably, the accuracy of mmWave technology-based positioning appears to be



closely linked to the distance from the target. In a study mentioned in (Ojas Kanhere, 2018), the authors conduct Angle of Arrival (AoA) and signal measurements in a  $35m \times 65.5m$  open space, achieving position accuracy ranging from  $16cm$  to  $3.25m$ . While positioning research using this mmWave technology is in its early stages, early theoretical findings and practical experiments reveal its potential to deliver the high accuracy demanded by modern smart applications. In a different context, (Hao, et al., 2022) introduces a multipath-assisted localization (MAL) model based on mmWave radar for indoor electronic device localization. This model effectively incorporates multipath effects when describing reflected signals, enabling precise target position determination using the MAL area formed by the reflected signal. Importantly, this model can provide 3D target information even when traditional Single-Input Single-Output (SISO) radar falls short. Furthermore, in a scenario described in (Y. Jia, 2018), an indoor office setting with a single mmWave base station (BS) is considered. The authors propose a method that fuses user equipment (UE) motion features, mmWave line-of-sight (LoS), and first-order reflection paths' Angle of Arrival (AoA) and Time of Arrival (ToA) for indoor positioning. They present an improved least mean square (LMS) algorithm to refine multipath AoA estimation and a modified multipath unscented Kalman filter (UKF) for position tracking. The results of these methods show significant enhancements in LoS-AoA estimation and centimeter-level 3D positioning accuracy, around  $60cm$ . Notably, this strategy is effective even in scenarios with insufficient anchor nodes. In an alternative approach, as presented in (Youqing Wang, 22), a method for achieving 3D indoor positioning using a single base station is proposed. This approach leverages multipath channels, with MIMO antennas estimating the angles of multipath coherent signals, and OFDM signals handling delay estimation. By integrating MIMO and OFDM technologies within a wireless communication system, an array antenna is employed to estimate the AoA of multipath signals. Spatial smoothing algorithms are applied in the frequency domain to estimate the Time Difference of Arrival (TDoA) of multiple coherent signals. This approach has been validated through simulations in a  $6m \times 8m \times 4.5m$  indoor space. The results indicate that positioning accuracy reaches submeter levels in 95% of cases and is less than  $0.4m$  in 60% of cases. The authors of (Z. Lin, 2018), present a novel 3-D indoor positioning scheme using mmWave massive multiple-input multiple-output (mMIMO) systems. The operation of this scheme is based on a hybrid received signal strength and angle of arrival (RSS-AoA) positioning scheme, which employs only a single access point equipped with a large-scale uniform cylindrical array. The authors design a novel hybrid RSS-AoA positioning scheme for the computations of the 3-D coordinates of the target mobile terminal. They demonstrate that their approach achieves azimuth and elevation precision around 0.5 degrees depending on the quality of the received signal. In (Tuo Wu, 2022), the authors investigate a 3D positioning algorithm for a mmWave system leveraging Reconfigurable Intelligent Surfaces (RIS) to enhance the positioning performance of mobile users (MUs). They use a two-stage weight least square (TSWLS) algorithm to obtain the closed-form solution of the MU's position. Similarly in (Yin, 2021), the authors address the channel estimation for RIS-aided mmWave communication systems based on a localization method. They propose the concept of reflecting unit set (RUS) to improve the flexibility of RIS. The authors then propose a novel coplanar maximum likelihood-based (CML) 3D positioning method based on the RUS and derive the Cramer-Rao lower

bound (CRLB) for the positioning method. Furthermore, they develop an efficient positioning-based channel estimation scheme with low computational complexity. They demonstrate that cm-level accuracy can be achieved averaging around 5cm depending on the received signal quality. A 60GHz signal-based positioning and tracking system is discussed in (A. Antonucci, 2019), which effectively filters out multiple reflections and diffuse scattering, ensuring a high level of accuracy. Operating within a longitudinal range of 0.46m to 5.55m and a lateral span from 1.91m to 3.04m, the system determines the target's position through the calculation of the local centroid in the associated point cloud. Overall, the system achieves a plane positioning accuracy with a 99% confidence level and an error of approximately 30–40cm. In another work (Maisy Lam, 2023), the authors have presented a self-localization system for autonomous drones that utilizes a single millimeter-Wave anchor. The system leverages a novel dual polarized, dual modulated mmWave anchor and mmWave-IMU Fusion self-localization algorithm to ultimately achieve precise, high-speed 3D localization. The authors have demonstrated a median localization error of 7cm and a 90th percentile less than 15cm, even in NLOS scenarios. Another work positioning a drone is presented in (Hunukumbure, 2022) where the authors developed a 3GPP-compliant drone-based 3D indoor localization solution employing an integration of time-based and angle-based techniques to improve the situational awareness in emergency situations and support emergency services. They have managed to achieve a horizontal and vertical positional error 1.05m and 0.7m at 26GHz frequency. A similar work is presented in (F. Parralejo, 2023) where the authors propose a security system based on millimetre-wave radar, using a processing workflow based on machine learning techniques, achieving 99.32% accuracy and 99.54% F1 score. Another work utilizing machine learning is presented in (P. K. Rai, 2021), where a custom CNN model achieves an accuracy of 95%. (Marques, 2020) presents an active drone detection system that uses a millimeter wave radar mounted on a drone to estimate 3D position of the target drone using 2D measurements. The results indicated an average 3D positioning error of 2.17m.

#### 4.2.1 2-DOF Setup Results

The experimental setup is presented in Figure 119. Using these measurements 3D positioning estimation was conducted both using a two 3D multilateration and a triangulation approach. Ground-truth location precision is crucial for the validity of this work as it serves as the reference for evaluating the accuracy of the approach. While flying around the lab, the drone was instructed to hover at the particular points of interest and while hovering the precise location of the drone was determined using a laser distance measuring tool. Essentially, the laser tool was secured using a tripod at the location of the drone while the latter was hovering and distance measurements were taken to the horizontal and vertical walls of the lab as well as to the floor determining the precise  $x,y,z$  location of the drone.

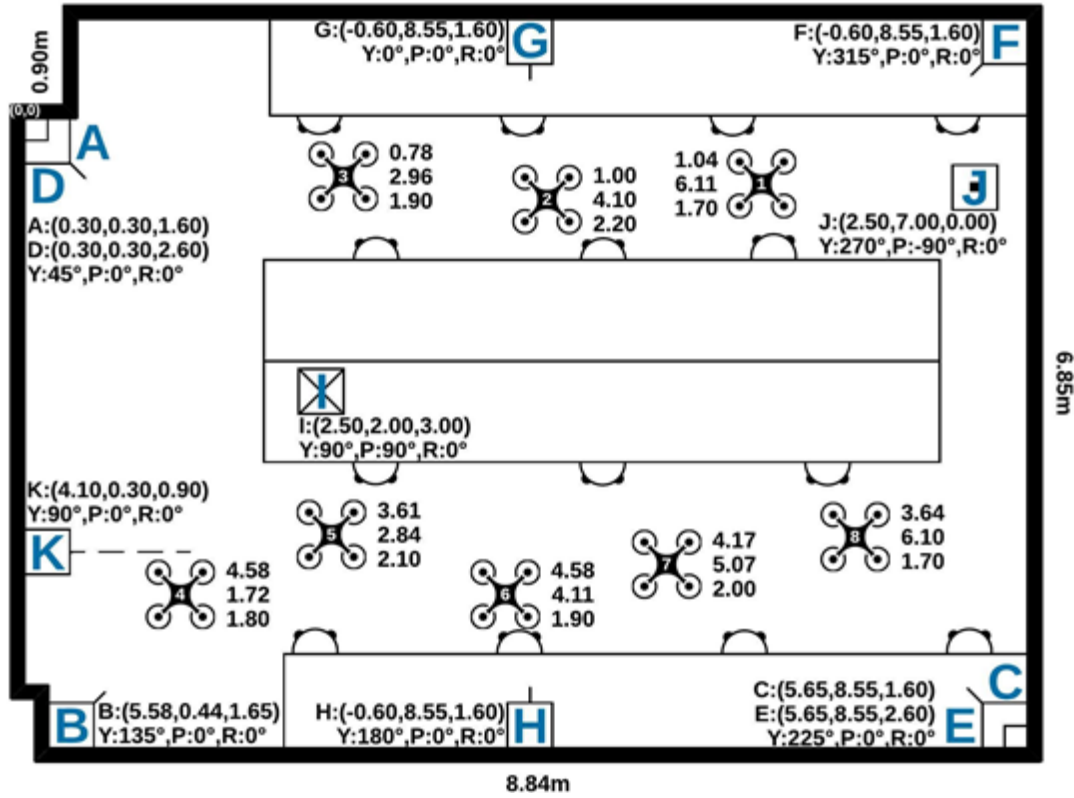


Figure 119: : mmWave 3D Positioning Experimental Setup using 2-DOF mmWave Sensors (Y: Yaw, P: Pitch, R: Roll)

#### 4.2.1 3D Multilateration Approach

Multilateration serves as a fundamental technique for achieving 3D positioning across a wide range of scientific and technological domains. It harnesses distance measurements from multiple reference points to determine the exact location of an object within three-dimensional space by using at least 4 sensors. Through the exploitation of geometric relationships between the object and these reference points, multilateration algorithms facilitate the calculation of intersecting spheres or hyperboloids, ultimately yielding the object's coordinates. In this work, 3D position estimation is done using the standard algebraic solution (Malivert F, 2023) of the 3D multilateration problem as well as a recursive multilateration solution presented in (Norrdine, 2012). Different combinations of sensors were used to investigate the effects of DoP.

The first experiment was conducted using ranging measurements collected from 4 TI sensors deployed in the 4 corners of the room in locations A, B, C and F as shown in **Error! Reference source not found.** and the results for both the standard algebraic solution as well as the recursive one are tabulated in Table 7. It appears that an average error in the ranging measurement of 0.06m translates into a 0.14m and 0.1m average positioning error in x and y using the standard algebraic solution while a considerable error is observed in the z-axis (5.72m average). These are translated into an average 3D positioning error of 5.76m. The results appear to improve when using the recursive multilateration approach

(vertical error of 1.69m and an average 3D error of 1.72m), however the error in the vertical dimension still remains significant. This is attributed to the fact that all sensors are placed on the same height resulting in a very high Vertical Dilution of Precision (VDOP) averaging around 23.6.

Table 7 - 4 Anchor Configuration - Equal Height

| Point   | Distance<br>Error (m) | Algebraic Solution |       |       |              | Recursive Solution |       |       |              | HDOP | VDOP  |
|---------|-----------------------|--------------------|-------|-------|--------------|--------------------|-------|-------|--------------|------|-------|
|         |                       | Error XYZ (m)      |       |       | 3D Error (m) | Error XYZ (m)      |       |       | 3D Error (m) |      |       |
|         |                       | x                  | y     | z     |              | x                  | y     | z     |              |      |       |
| 1       | -0.04                 | 0.12               | 0.23  | 0.08  | 0.27         | 0.02               | -0.27 | -1.02 | 1.05         | 1.12 | 43.05 |
| 2       | -0.06                 | 0.22               | 0.07  | -9.46 | 9.46         | 0.28               | -0.02 | -1.23 | 1.26         | 1.10 | 25.95 |
| 3       | 0.09                  | 0.12               | 0.00  | -1.35 | 1.35         | 0.09               | -0.12 | 0.36  | 0.39         | 1.11 | 4.95  |
| 4       | 0.03                  | -0.21              | -0.09 | 0.43  | 0.49         | -0.12              | -0.22 | 4.77  | 4.78         | 1.11 | 8.35  |
| 5       | -0.02                 | -0.22              | 0.12  | -16.4 | 16.40        | 0.02               | -0.26 | 2.39  | 2.40         | 1.06 | 4.95  |
| 6       | 0.10                  | -0.07              | 0.08  | -11.1 | 11.11        | 0.00               | -0.06 | -1.68 | 1.68         | 1.11 | 10.59 |
| 7       | -0.09                 | 0.05               | 0.06  | -6.56 | 6.56         | 0.39               | 0.09  | 1.54  | 1.59         | 1.24 | 44.70 |
| 8       | -0.02                 | -0.13              | -0.16 | 0.40  | 0.45         | -0.22              | 0.17  | -0.56 | 0.62         | 1.25 | 46.41 |
| Average | 0.06                  | 0.14               | 0.10  | 5.72  | 5.76         | 0.14               | 0.15  | 1.69  | 1.72         | 1.14 | 23.61 |

Table 8 - 4 Anchor Configuration - Different Height

| Point   | Distance<br>Error (m) | Algebraic Solution |       |       |              | Recursive Solution |       |       |              | HDOP | VDOP |
|---------|-----------------------|--------------------|-------|-------|--------------|--------------------|-------|-------|--------------|------|------|
|         |                       | Error XYZ (m)      |       |       | 3D Error (m) | Error XYZ (m)      |       |       | 3D Error (m) |      |      |
|         |                       | x                  | y     | z     |              | x                  | y     | z     |              |      |      |
| 1       | 0.07                  | -0.19              | -0.20 | -0.15 | 0.32         | -0.29              | 0.00  | -0.37 | 0.47         | 1.07 | 1.80 |
| 2       | 0.05                  | -0.09              | -0.25 | 0.03  | 0.27         | 0.28               | -0.31 | -0.12 | 0.44         | 1.06 | 1.61 |
| 3       | -0.14                 | 0.23               | 0.02  | 0.47  | 0.52         | 0.11               | -0.09 | -0.33 | 0.36         | 1.12 | 1.75 |
| 4       | -0.06                 | -0.13              | -0.29 | 0.12  | 0.34         | -1.59              | -0.02 | 1.94  | 2.51         | 1.31 | 1.49 |
| 5       | -0.06                 | 0.08               | 0.02  | -0.14 | 0.16         | -2.94              | -0.19 | 2.81  | 4.07         | 1.22 | 1.41 |
| 6       | -0.12                 | 0.21               | 0.07  | 0.03  | 0.22         | -1.53              | -0.27 | 1.96  | 2.50         | 1.07 | 1.03 |
| 7       | 0.13                  | 0.06               | -0.22 | 0.04  | 0.23         | -0.39              | -0.17 | 0.09  | 0.44         | 1.10 | 0.93 |
| 8       | -0.07                 | -0.29              | 0.23  | -0.11 | 0.38         | 0.13               | 0.27  | -0.33 | 0.45         | 1.06 | 1.16 |
| Average | 0.09                  | 0.16               | 0.16  | 0.13  | 0.31         | 0.91               | 0.17  | 0.99  | 1.40         | 1.41 | 1.54 |

Table 9 - 6 Anchor Configuration - Different Height

| Point   | Distance<br>Error (m) | Algebraic Solution |       |       |              | Recursive Solution |       |       |              | HDOP | VDOP |
|---------|-----------------------|--------------------|-------|-------|--------------|--------------------|-------|-------|--------------|------|------|
|         |                       | Error XYZ (m)      |       |       | 3D Error (m) | Error XYZ (m)      |       |       | 3D Error (m) |      |      |
|         |                       | x                  | y     | z     |              | x                  | y     | z     |              |      |      |
| 1       | 0.00                  | -0.18              | 0.07  | -0.56 | 0.59         | -0.07              | 0.01  | -0.14 | 0.15         | 0.93 | 1.60 |
| 2       | -0.08                 | 0.11               | -0.05 | -0.27 | 0.30         | -0.12              | -0.27 | -0.06 | 0.30         | 0.95 | 1.63 |
| 3       | 0.04                  | -0.02              | -0.03 | 0.17  | 0.17         | 0.64               | 0.01  | 0.67  | 0.93         | 0.93 | 1.70 |
| 4       | -0.12                 | -0.04              | 0.11  | -0.12 | 0.17         | 0.01               | -0.34 | -0.05 | 0.35         | 0.96 | 1.60 |
| 5       | -0.02                 | -0.05              | 0.08  | 0.07  | 0.11         | 0.52               | -0.15 | 0.03  | 0.55         | 0.93 | 1.44 |
| 6       | -0.07                 | 0.07               | -0.03 | 0.07  | 0.10         | -0.01              | -0.14 | -0.31 | 0.34         | 0.88 | 1.74 |
| 7       | 0.13                  | -0.08              | -0.02 | 0.48  | 0.49         | 0.08               | 0.05  | 0.19  | 0.22         | 0.91 | 1.68 |
| 8       | -0.11                 | 0.03               | 0.00  | -0.01 | 0.03         | 0.24               | 0.10  | 0.10  | 0.28         | 0.95 | 1.45 |
| Average | 0.07                  | 0.07               | 0.05  | 0.22  | 0.24         | 0.21               | 0.13  | 0.19  | 0.39         | 0.94 | 1.48 |

DOP plays a crucial role in 3D indoor positioning, as it directly affects the accuracy and reliability of position estimates. While DOP values are commonly considered in the horizontal plane, they are equally important in the vertical plane (B. Li, 2020). Considering this, the reason why our results often exhibit better accuracy in the horizontal plane compared to the vertical plane can be attributed to the distribution of sensors. In the horizontal plane, the sensors are spread out more widely, allowing for better sensor geometry. This improved distribution of sensors results in lower HDOP values, indicating reduced potential for horizontal positioning errors. The IWR1642BOOST mmWave sensor, with its narrow 15-degree elevation field-of-view, poses a limitation on the distribution of sensors in the vertical plane. The narrower vertical perspective leads to a less favorable sensor geometry and higher VDOP values. As a consequence, the accuracy of height estimation in 3D positioning may be more susceptible to errors and uncertainties. Nevertheless, it was attempted to position 4 sensors at different heights (1m, 1.5m, 2m and 2.5m) to demonstrate the potential improvement. Table 8 verifies this hypothesis by indicating significant improvement of algebraic solution in the z-axis (0.13m) bringing the 3D positioning error down to 0.31m which is attributed to the significant improvement of the VDOP (average 1.54). Interestingly enough it appears that the recursive solution fails to identify the optimal solution leading to significantly high errors. To further investigate the DOP significance we set up another experiment consisting of 6 anchors (the four anchors of the previous case plus one sensor at the ceiling (position I) and one on the floor (position J)). This new constellation or anchors reduces both the VDOP as well as the HDOP and this reflected on both the multilateration approaches. The average 3D positioning error reduces down to 0.22m while the one from the recursive version reduces down to 0.39m. Interestingly the standard algebraic solution still outperforms the recursive one as can be seen in Table 9.



#### 4.2.2 3D Triangulation Approach

Considering the inaccuracy of the multilateration approach in the z-axis, particularly when DOP optimization is not possible, and capitalizing on the ability of the IWR1642BOOST sensor to measure the azimuth angle, the experimental setup was adjusted, deploying 2 sets of two sensors on top of each other as shown in Figure 100. Sensor **D** is placed on top of **A**, sensor **E** on top of **C**, while sensor **F** was left on its own on the far-most right corner. 3D position estimation is achieved by using a combination of typical triangulation formulation using the azimuth angles measured from the 3 corners while the z-axis coordinate is estimated based on the height formulation below which estimates the height  $h$  in the Complexity-Reduced Trilateration Approach (COLA) approach presented in (Marrón, 2010).

$$h = z_2 - \frac{d_2^2 - d_1^2 + (z_2 - z_1)^2}{2(z_2 - z_1)}$$

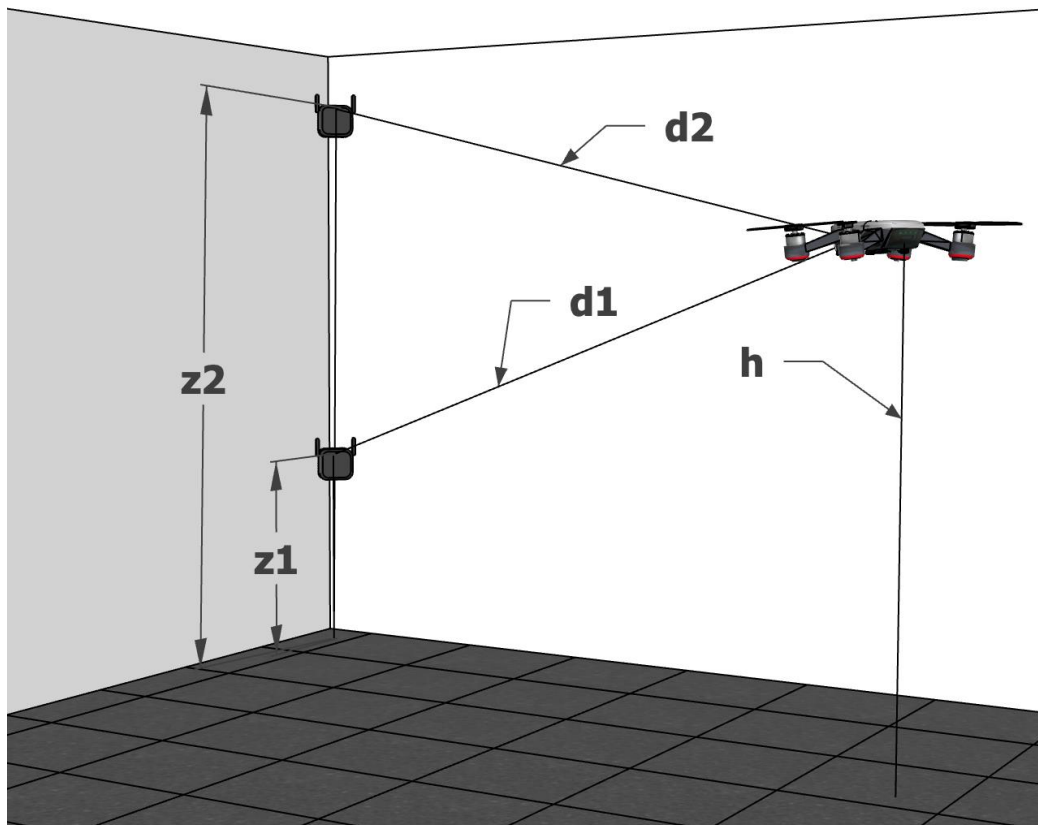


Figure 120 - 3D Triangulation - Sensor Arrangement

Results tabulated in Table 10 indicate a significant improvement in the z-axis (0.11m) while there is also a good improvement in the x and y axes (error being 0.09m and 0.08m) bringing the 3D positioning accuracy down to 0.17m.

Table 10 - 3D Triangulation Positioning

| Point          | Azimuth Error (°) | XYZ Error (m) |             |             | 3D Error (m) |
|----------------|-------------------|---------------|-------------|-------------|--------------|
|                |                   | x             | y           | z           |              |
| 1              | 1.02              | 0.13          | 0.14        | -0.03       | 0.20         |
| 2              | 0.72              | -0.03         | 0.09        | 0.16        | 0.19         |
| 3              | 0.91              | -0.08         | -0.09       | 0.14        | 0.19         |
| 4              | 1.88              | -0.04         | -0.13       | 0.16        | 0.22         |
| 5              | 1.51              | 0.25          | 0.11        | -0.13       | 0.31         |
| 6              | 0.91              | 0.09          | -0.08       | -0.30       | 0.32         |
| 7              | 0.39              | 0.06          | 0.03        | 0.04        | 0.08         |
| 8              | 2.20              | 0.05          | -0.02       | -0.03       | 0.06         |
| <b>Average</b> | <b>1.20</b>       | <b>0.09</b>   | <b>0.08</b> | <b>0.11</b> | <b>0.17</b>  |

As it can be seen from Table 10, the average 3D error using mmWave radar sensors is 17 cm with a maximum reaching 32 cm and a minimum being at 6 cm. The results are very comparable to the results obtained from the cameras. An advantage the cameras offer over the mmWave technology is that the object can not only be detected but also identified. However, custom made navigation controllers are required in order to send the individual location to the individual identified object.

In conclusion, for indoor greenhouse applications both mmWave and camera technologies are acceptable solutions. Even the maximum error of 35 cm satisfied the set KPI which was aiming at a sub-meter accuracy. Hence, the KPI requirements have been successfully met.

## 5 Indoor Positioning by GPS Conversion and Retransmission

It is well known that GPS is for outdoor applications. For indoor or covered areas where the signal is obstructed then the GPS signal is blocked and GPS based systems do not work. The proposed greenhouse application fall in the same category of indoor and covered areas where the GPS signals are blocked and do not work.

This research is proposing a system where unmanned systems, UGV and UAV, which will navigate in the greenhouse, will be detected by a camera system or mmWave radar sensors and then cameras will calculate their local location within the greenhouse. When the local coordinates are calculated then they will be converted to GPS coordinate system and retransmitted. This is also known as “GPS spoofing: The main reason this research is proposing retransmission of GPS signals is so that commercially available controllers and autopilots will still be operational receiving GPS navigation. For example, a commercially available UAV such as the DJI Mavic 3M will still navigate because the GPS signal will be available.

On the other hand, following the success of the experimental results described and analysed in section 4 (using image processing for detection and mmWave radar sensors and cameras for Indoor 3D positioning) this research identifies a drawback which opens room for future research and improvement. The fact that this research identified the capability of detecting and identifying simultaneously multiple unmanned systems, then this research identifies the possibility that GPS spoofing can work for multiple unmanned systems simultaneously at different indoor locations for as long as they have custom made controllers. This is future work that this team will undertake due to the promising and successful outcomes of this work.

### 5.1 Experimental Setup

The experimental setup consisted of a commercially available autopilot, Ardupilot AMP2.5, a Software Defined Radio (SDR), HackRF One and a laptop.

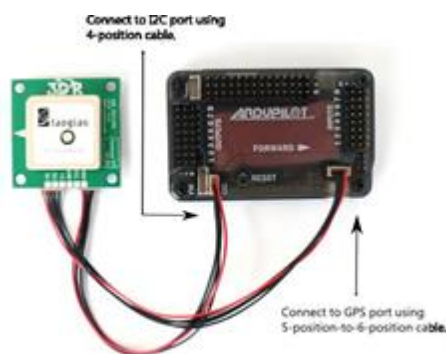


Figure 121: Ardupilot with GPS module connected



Figure 122: 2 HackRF ONE

Additionally the following software were required:

- GPS-SDR-SIM: <https://github.com/osqzss/gps-sdr-sim> (needs to be compiled in visual studio to run)
- Visual studio: <https://visualstudio.microsoft.com/vs/community/>
- Mission Planner: <https://ardupilot.org/planner/docs/mission-planner-installation.html>
- PothosSDR: <http://downloads.myriadrf.org/builds/PothosSDR/>
- SatGen V3: <https://www.labsat.co.uk/index.php/en/customer-area/software-firmware>
- Latest RINEX navigation ephemerides broadcast brdc file from Nasa. An account is required which can be created free: <https://cdis.nasa.gov/archive/gnss/data/daily/2023/brdc/>

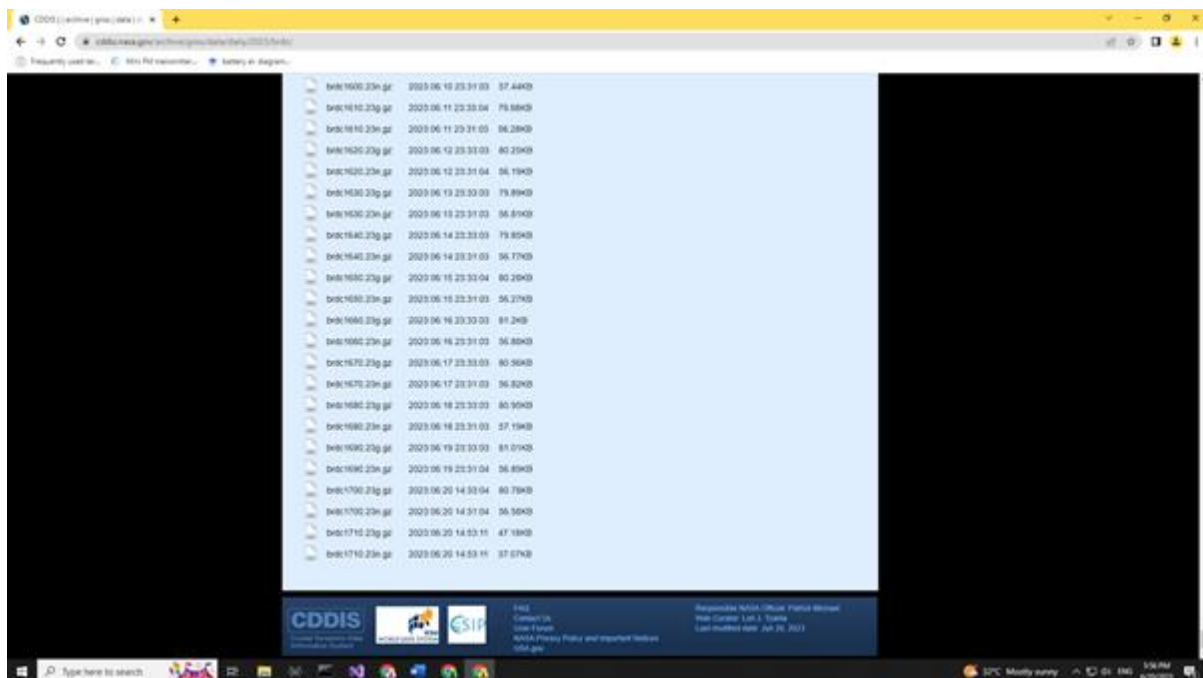


Figure 123: Latest ephemeris broadcast file

### 5.1.1 STATIC GPS SPOOFING

#### STEP 1: Build the gps-sdr-sim.exe

After you install all the software listed above a custom executable file needs to be compiled following the below instructions.

Windows build instructions

1. Start Visual Studio.
2. Create an empty project for a console application.
3. On the Solution Explorer at right, add "gpssim.c" and "getopt.c" to the Source Files folder.

4. Select "Release" in Solution Configurations drop-down list.
5. Build the solution.

## STEP 2: Organize your files.

Create a new folder named "gps instructions" at desktop and add the gps-sdr-sim.exe compiled file from above and the updated brdc file.

## STEP 3: Run mission planner.

Run mission planner and connect ARDUPILLOT to the current COM port.

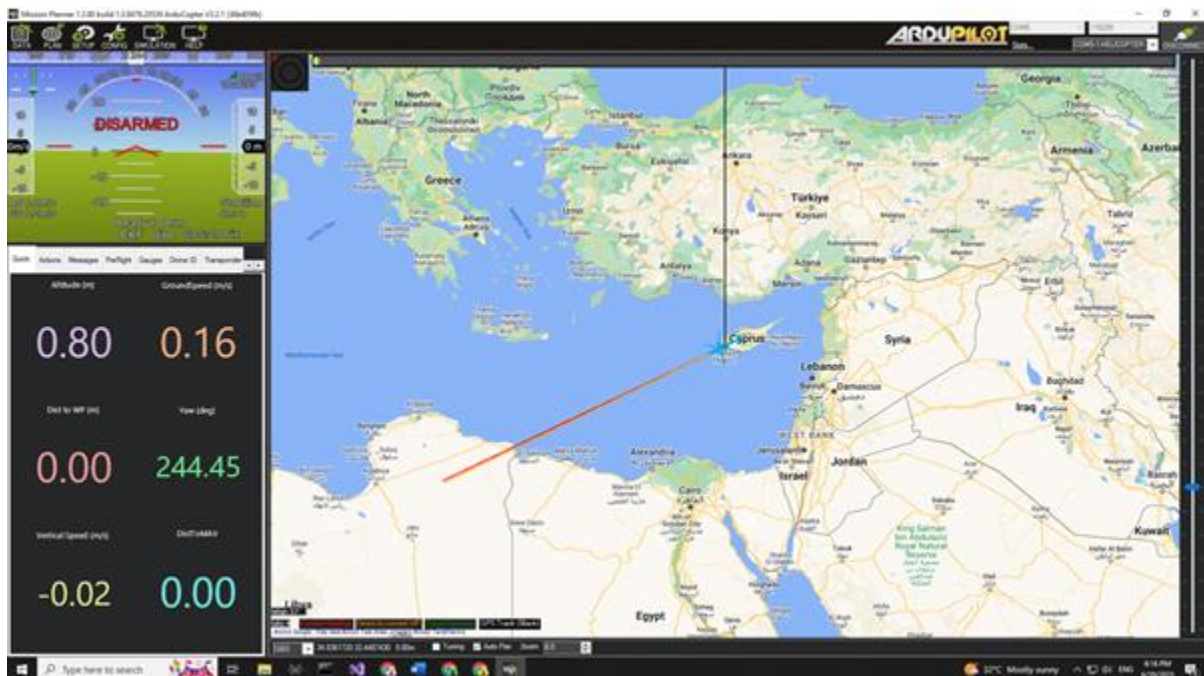


Figure 124: GPS Signal before transmission

## STEP 4: Generate the GPS bin file.

- Open Command Prompt and change directory folder to the "gps instructions"
- Run this command and a bin file will be generated in the gps instructions folder:

• `gps-sdr-sim -e brdc3540.14n -l 30.286502,120.032669 -b 8`

*file name of daily efemerides file*

*\*different filename everyday*

*Coordinates of static location for spoofing signal*



```

Command Prompt
Microsoft Windows [Version 10.0.19045.3086]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Demetris>cd "C:\Users\Demetris\Desktop\gps instructions"

C:\Users\Demetris\Desktop\gps instructions>gps-sdr-sim.exe -e brdc1700.23n -l 35.1698,33.4366 -b 8
Using static location mode.
xyz = 4355734.7, 2876071.7, 3653282.0
llh = 35.169800, 33.436600, 0.0
Start time = 2023/06/19,00:00:00 (2267:86400)
Duration = 300.0 [sec]
05 230.9 16.5 24117189.2 3.5
07 96.9 8.0 24690146.7 4.2
13 287.7 78.5 20255963.9 1.5
14 40.8 51.5 21307121.6 1.8
15 301.9 44.6 21428813.8 2.0
20 208.7 4.2 25206301.2 4.6
23 322.4 3.1 25429533.5 4.7
24 295.7 11.5 24177689.1 3.9
30 85.6 34.4 22367139.0 2.4
Time into run = 300.0
Done!
Process time = 16.0 [sec]

C:\Users\Demetris\Desktop\gps instructions>

```

Figure 125: Command to generate the bin file

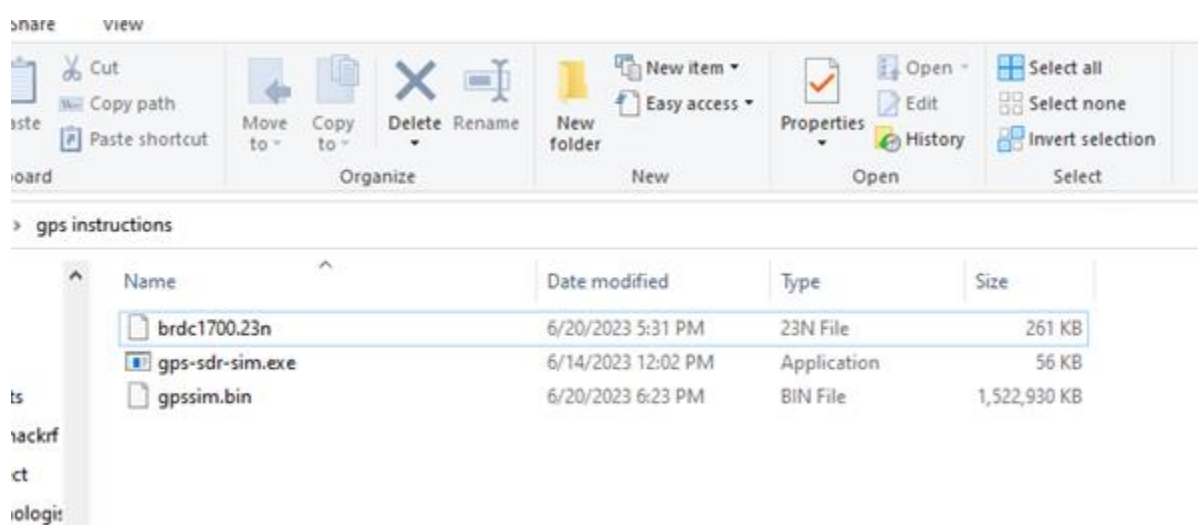
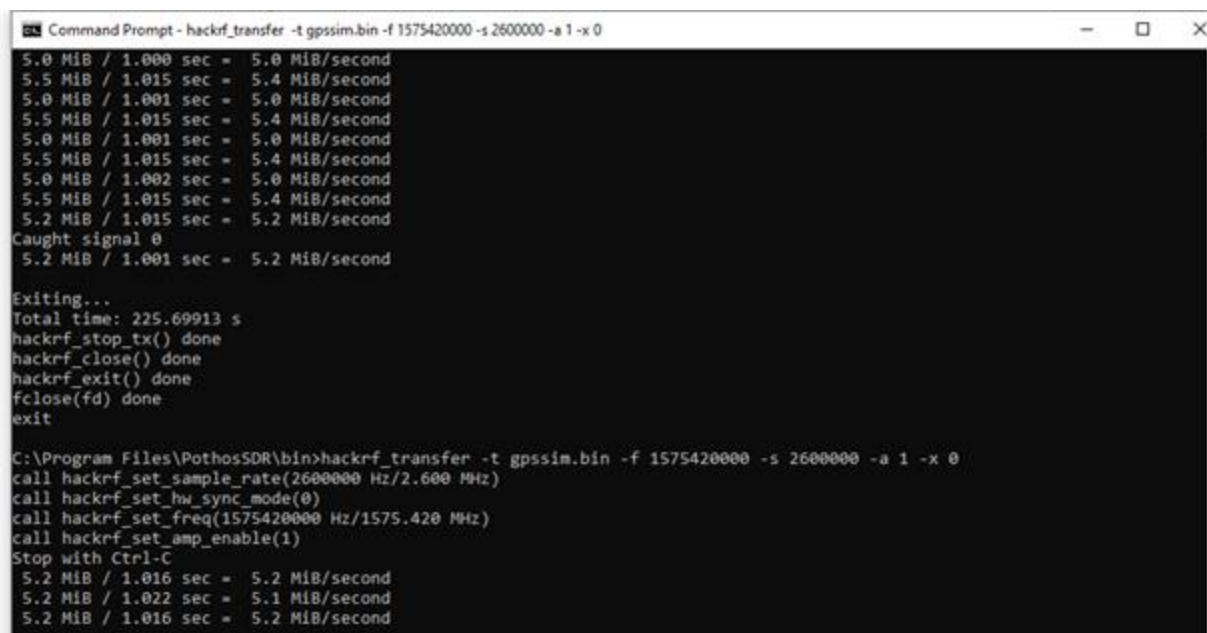


Figure 126: folder organization

#### STEP 5: Transmit the GPS bin file.

1. Connect HACKRF to pc and fix the antenna to a desired location-near the Ardupilot (approximately 80 cm away).
2. Search the file "hackrf\_transfer" in the pc and open the file location

3. Cut the gpssim.bin file generated above and paste to the same folder that the "hackrf\_transfer" file is located ("PothosSRD >> bin" )
4. Open the command prompt and locate the above directory.
5. Run this command: **hackrf\_transfer -t gpssim.bin -f 1575420000 -s 2600000 -a 1 -x 0**
6. After approximately 30 seconds the GPS module will start receiving the signal. The blue led light on the module will remain ON indicating GPS FIX. The new static location will appear on mission planner.



```

Command Prompt - hackrf_transfer -t gpssim.bin -f 1575420000 -s 2600000 -a 1 -x 0
5.0 MiB / 1.000 sec = 5.0 MiB/second
5.5 MiB / 1.015 sec = 5.4 MiB/second
5.0 MiB / 1.001 sec = 5.0 MiB/second
5.5 MiB / 1.015 sec = 5.4 MiB/second
5.0 MiB / 1.001 sec = 5.0 MiB/second
5.5 MiB / 1.015 sec = 5.4 MiB/second
5.0 MiB / 1.002 sec = 5.0 MiB/second
5.5 MiB / 1.015 sec = 5.4 MiB/second
5.2 MiB / 1.015 sec = 5.2 MiB/second
Caught signal 0
5.2 MiB / 1.001 sec = 5.2 MiB/second
Exiting...
Total time: 225.69913 s
hackrf_stop_tx() done
hackrf_close() done
hackrf_exit() done
fclose(fd) done
exit

C:\Program Files\PothosSDR\bin>hackrf_transfer -t gpssim.bin -f 1575420000 -s 2600000 -a 1 -x 0
call hackrf_set_sample_rate(2600000 Hz/2.600 MHz)
call hackrf_set_hw_sync_mode(0)
call hackrf_set_freq(1575420000 Hz/1575.420 MHz)
call hackrf_set_amp_enable(1)
Stop with Ctrl-C
5.2 MiB / 1.016 sec = 5.2 MiB/second
5.2 MiB / 1.022 sec = 5.1 MiB/second
5.2 MiB / 1.016 sec = 5.2 MiB/second

```

Figure 127: Command for bin file transmission

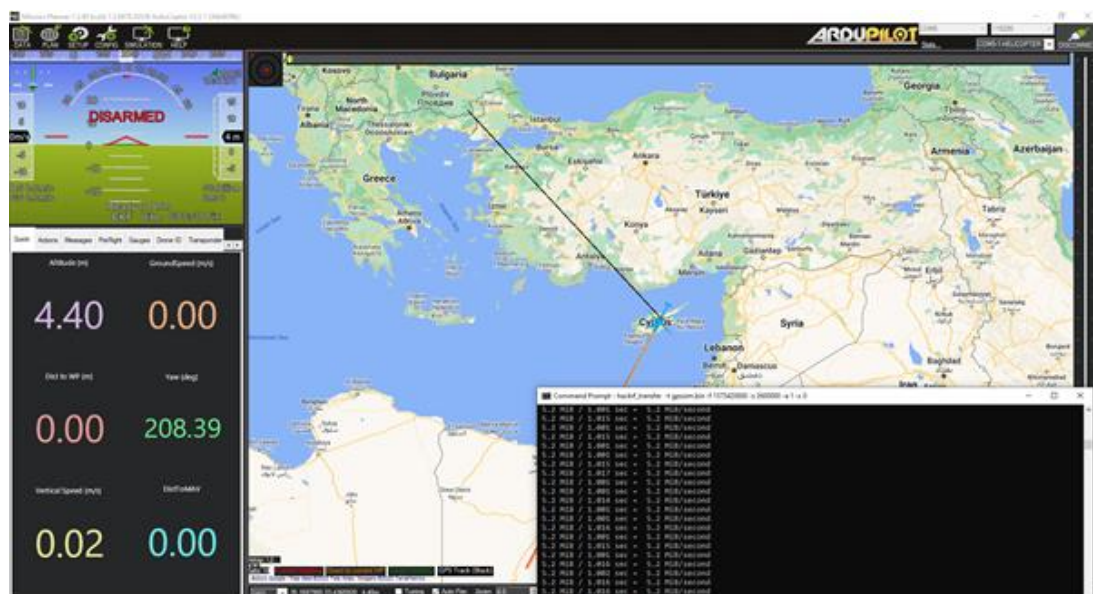


Figure 128: Mission planner after GPS transmission

### 5.1.2 DYNAMIC GPS SPOOFING

Before proceeding further with dynamic spoofing first test a static location as instructed above since some actions and procedures are overlapping.

#### STEP 1: Draw a route with SatGen

1. The demo version of SatGen enables the generation of 120 seconds dynamic GPS signal.
2. At the draw route tab, browse google maps to find the desired location and draw a route to follow or upload a csv file with a certain route.

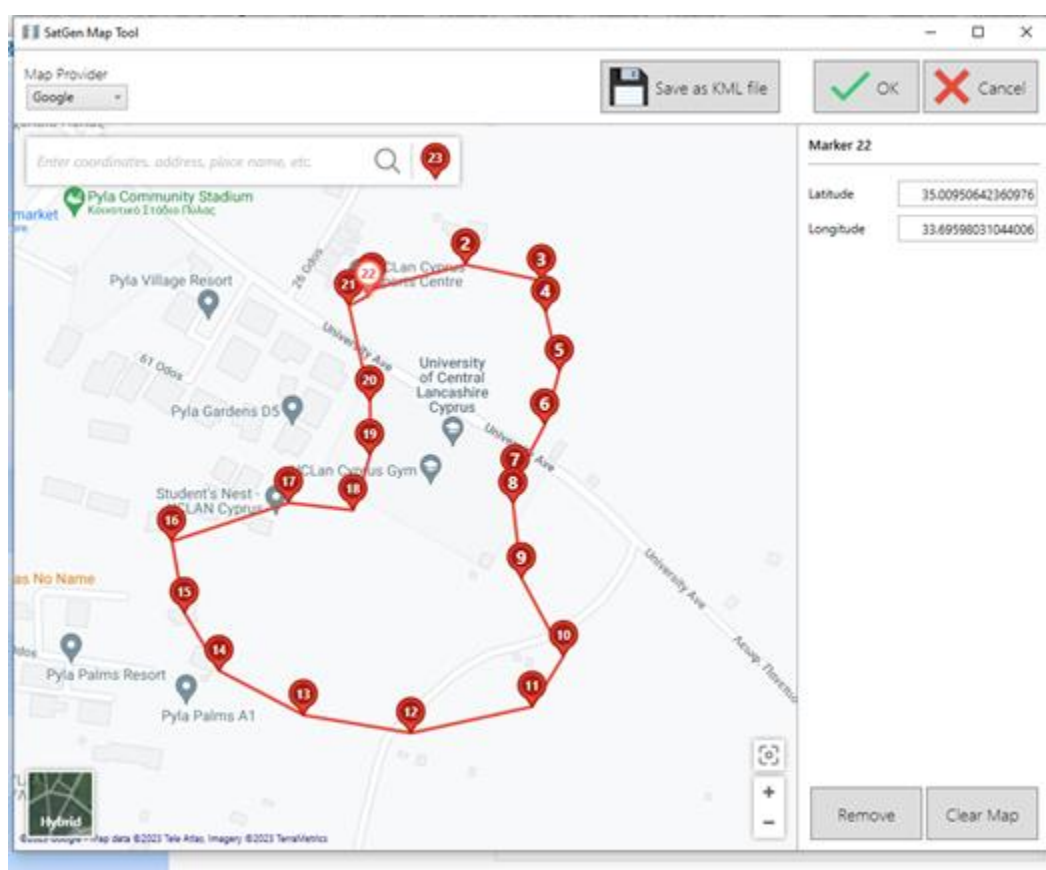


Figure 129: Draw a custom route from google maps provider in SatGen

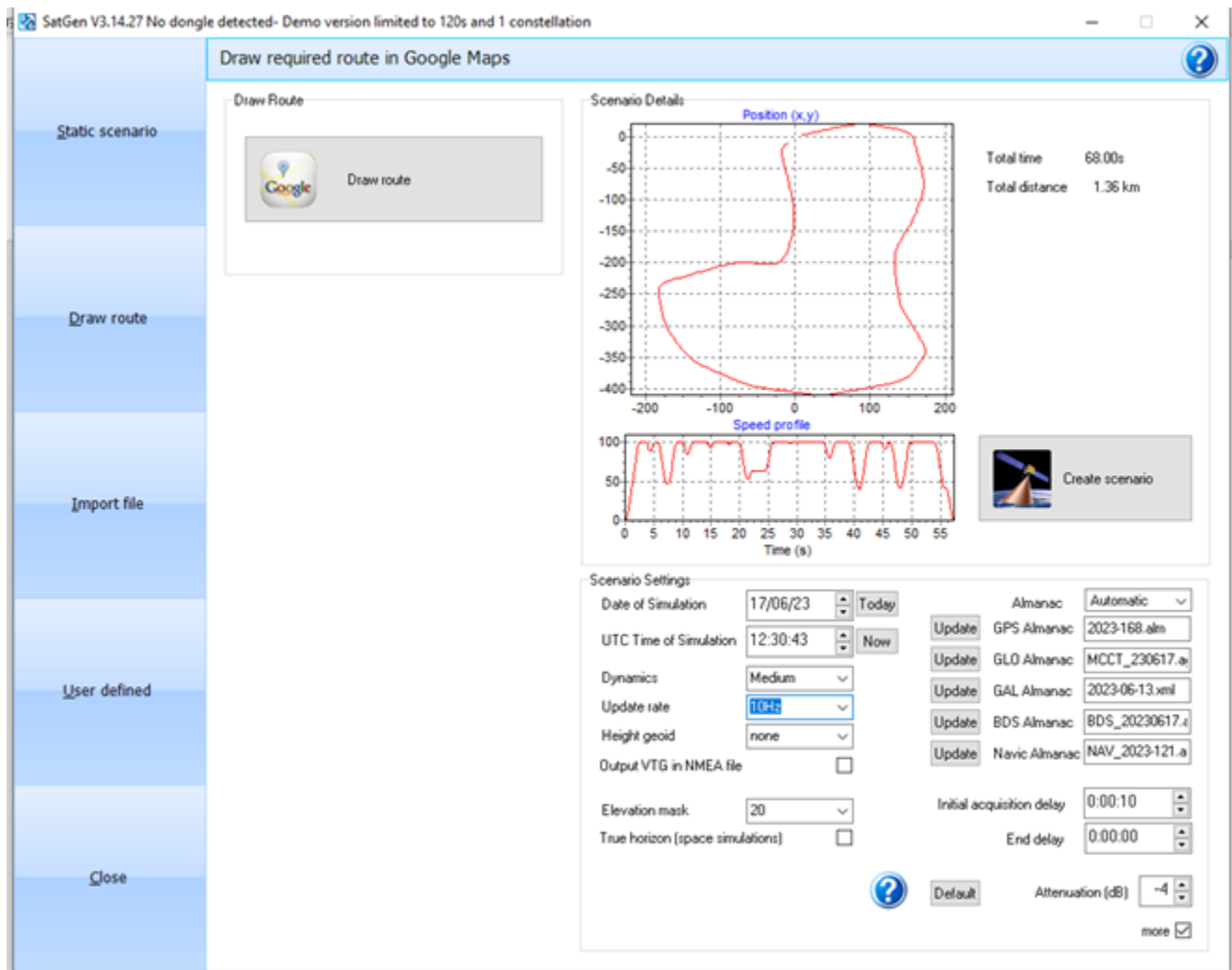


Figure 130: Parameters for dynamic scenario

## STEP 2: Adjust desired parameters.

- Velocity
  - High Dynamics will have a maximum speed of 300 km/h, Medium Dynamics has a maximum of 100 km/h, Low Dynamics has a maximum of 60 km/h and in User Dynamics the maximum speed can be dictated by the user. The software will automatically smooth the dynamic data to reflect turns at slower speeds.
- Update rate
  - You can define the update rate for the route from 1 Hz to 100 Hz. This will define the granularity of the route created in the software.
- Initial acquisition delay
  - The initial acquisition delay should be set to reflect the static time allowed for the device under test (DUT) to acquire the signals and to start to navigate. Set to 10-30 seconds.

- After you finish with all parameters, click create scenario >> leave default settings at next window and click ok.

### STEP 3: Copy the KML file.

- A window will pop-up indicating the folder with generated files. Copy KMLoutput.txt to “gps instructions” folder.

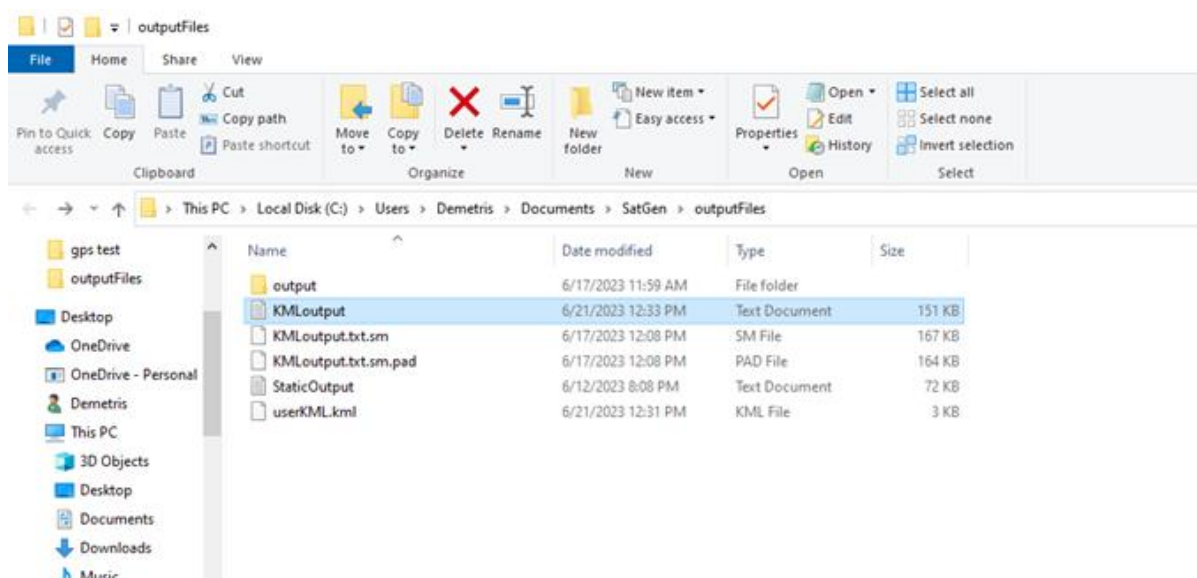


Figure 131: Dynamic KML file

### STEP 4: Generate dynamic BIN file for transmission.

- Open command prompt and locate gps instructions directory.
- Run this command **gps-sdr-sim -e brdc1720.23n -g KMLoutput.txt -b 8**
- Copy and replace gpssim.bin file to the directory of hackrf\_transfer.exe

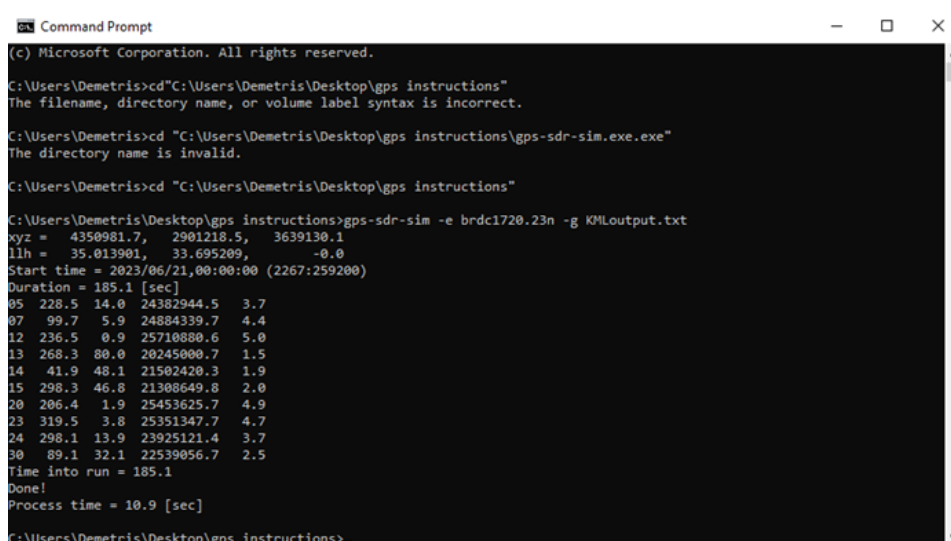


Figure 132: Dynamic bin file generation



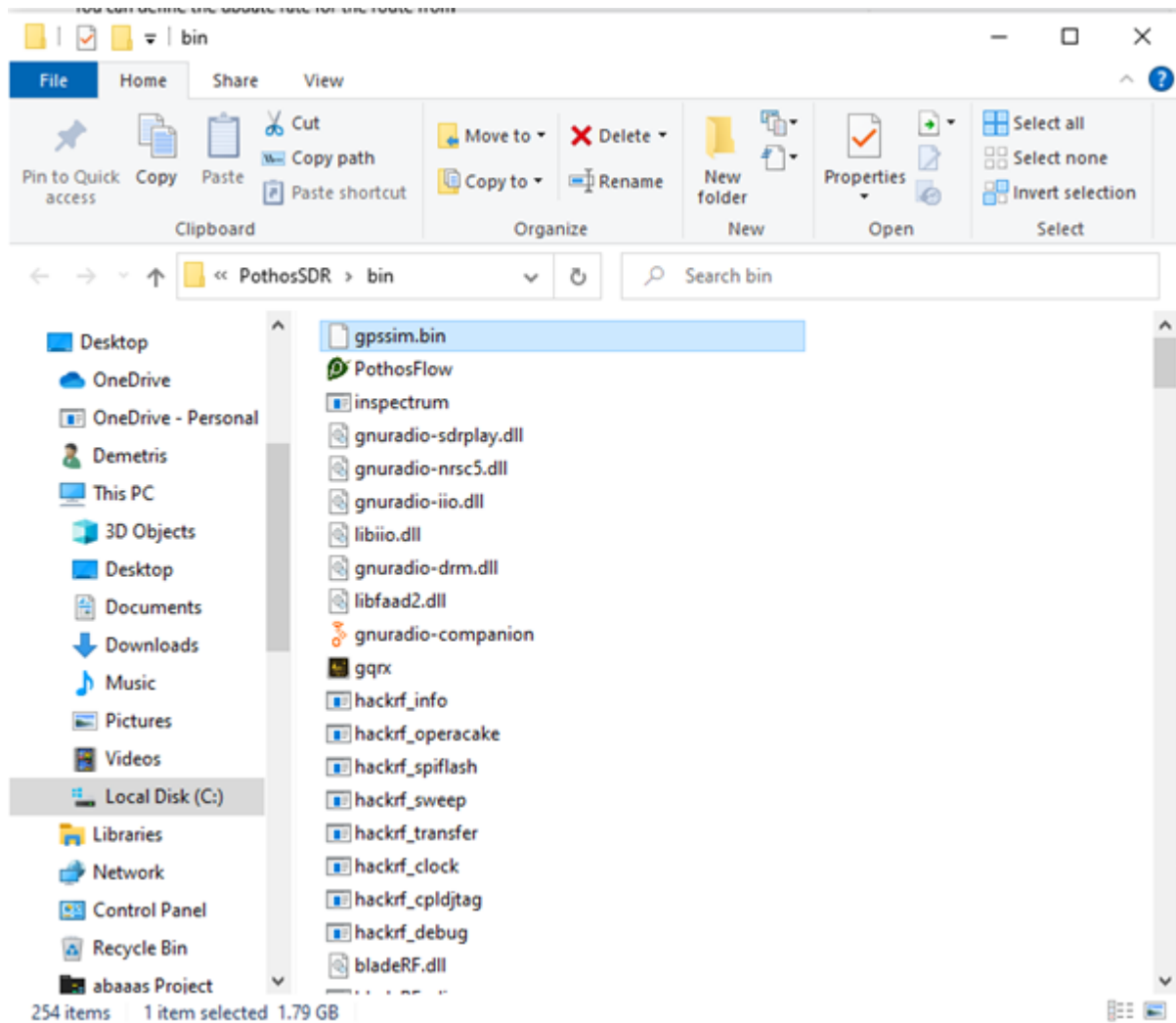


Figure 133: add gpssim.bin to same directory with hackrf\_transfer

#### STEP 5: Transmit the dynamic BIN file.

1. Open again command prompt and change directory to the above directory (i.e. directory of hackrf\_transfer and gpssim.bin)
2. Run this command: `hackrf_transfer -t gpssim.bin -f 1575420000 -s 2600000 -a 1 -x 0`
3. After approximately 30 seconds the GPS receiver will receive the signal and blue led will remain ON during the transmission. Mission planner will start to show the route displayed by the moving vehicle.

```

Command Prompt

exit

C:\Program Files\PothosSDR\bin>hackrf_transfer -t gpssim.bin -f 1575420000 -s 2600000 -a 1 -x 0
call hackrf_set_sample_rate(2600000 Hz/2.600 MHz)
call hackrf_set_hw_sync_mode(0)
call hackrf_set_freq(1575420000 Hz/1575.420 MHz)
call hackrf_set_amp_enable(1)
Stop with Ctrl-C
5.2 MiB / 1.016 sec = 5.2 MiB/second
5.2 MiB / 1.014 sec = 5.2 MiB/second
5.2 MiB / 1.007 sec = 5.2 MiB/second
5.2 MiB / 1.015 sec = 5.2 MiB/second
5.2 MiB / 1.016 sec = 5.2 MiB/second
5.2 MiB / 1.015 sec = 5.2 MiB/second
5.2 MiB / 1.016 sec = 5.2 MiB/second
5.2 MiB / 1.015 sec = 5.2 MiB/second
5.2 MiB / 1.016 sec = 5.2 MiB/second
5.5 MiB / 1.015 sec = 5.4 MiB/second
5.2 MiB / 1.015 sec = 5.2 MiB/second
5.2 MiB / 1.015 sec = 5.2 MiB/second
5.2 MiB / 1.015 sec = 5.2 MiB/second
5.2 MiB / 1.015 sec = 5.2 MiB/second
5.2 MiB / 1.015 sec = 5.2 MiB/second
5.2 MiB / 1.015 sec = 5.2 MiB/second
5.2 MiB / 1.015 sec = 5.2 MiB/second
5.5 MiB / 1.015 sec = 5.4 MiB/second
5.2 MiB / 1.015 sec = 5.2 MiB/second
5.2 MiB / 1.015 sec = 5.2 MiB/second
5.2 MiB / 1.015 sec = 5.2 MiB/second
5.2 MiB / 1.015 sec = 5.2 MiB/second

```

Figure 134: HackRF dynamic transmission prompt

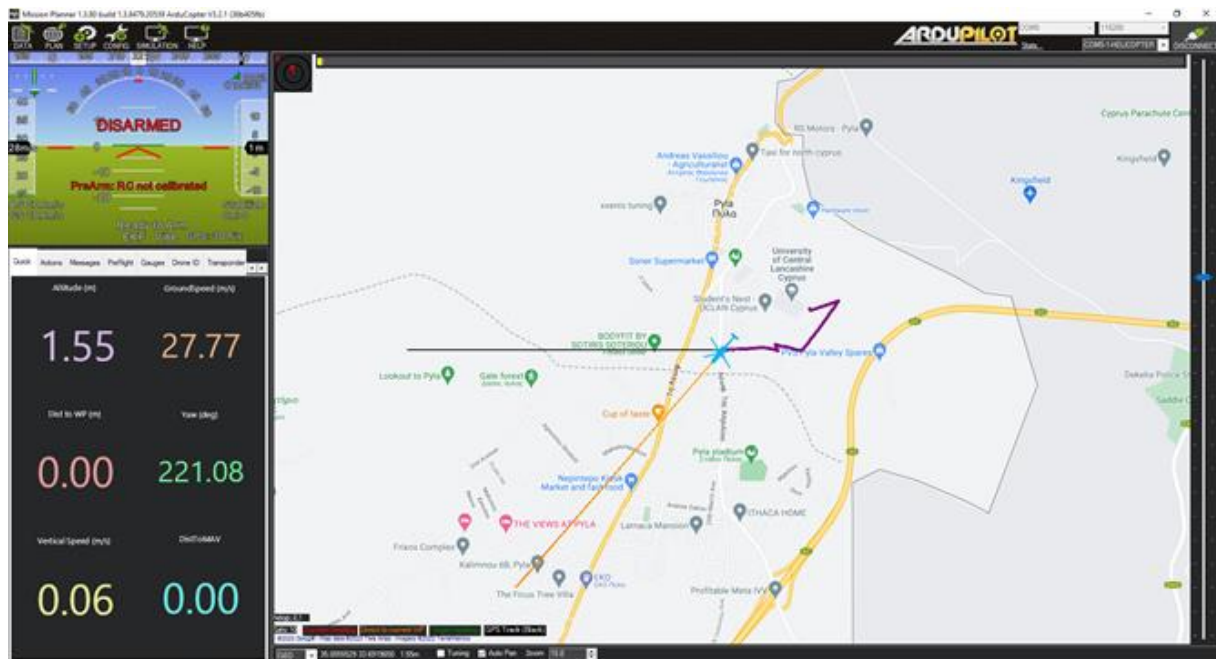


Figure 135: Mission planner during dynamic GPS transmission

## 5.2 Experimental Results

Google Earth was used to identify the GPS coordinates at the Electronics Laboratory at University UCLan Cyprus as shown on Figure 136. Furthermore, on Figure 137 it is clearly evident that in the laboratory there is not GPS signal hence the mission planner shows that the autopilot is “Disarmed” whereas after the transmission as shown on Figure 138 the autopilot is shown as “Armed” indicating the presence of strong GPS signal. In addition worth noting that the mission planner identifies the GPS coordinates as being in Cyprus at Pyla, Larnaca.



Figure 136: GPS Coordinates UCLan Cyprus from Google Earth



Figure 137: Mission Planner Before GPS Transmission





Figure 138: Mission Planner After GPS Transmission

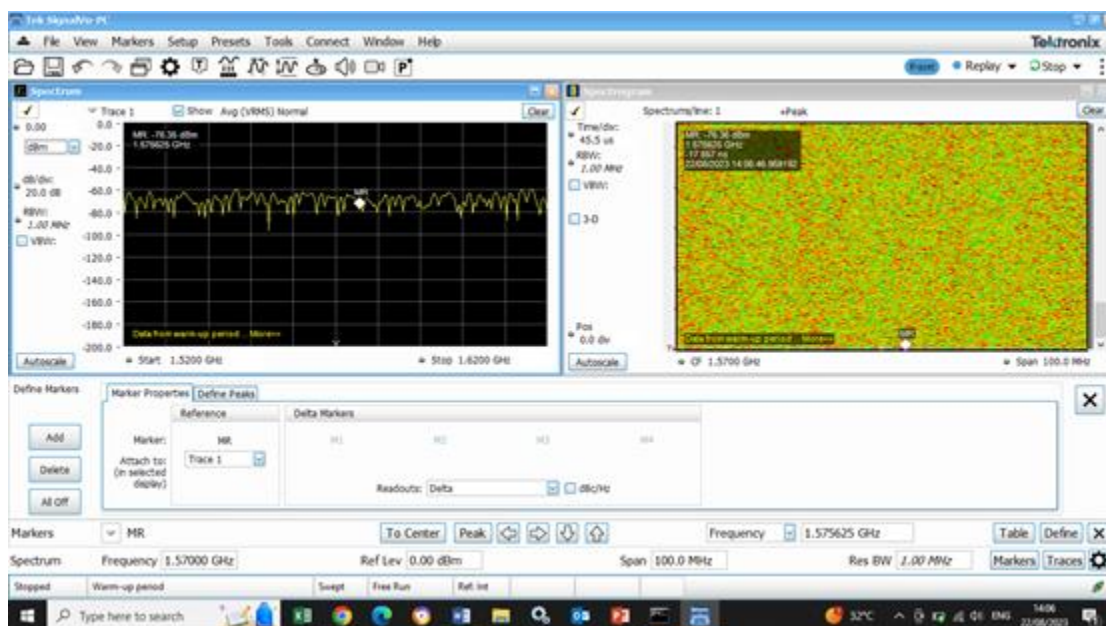


Figure 139: Spectrum before GPS Transmission

Figure 139 shows the monitoring of GPS signals inside the laboratory using a spectrum analyser. As it can be seen there are not any GPS signals presents. On the other hand as soon as transmission starts then on Figure 140 it can be seen that a GPS signal is present.

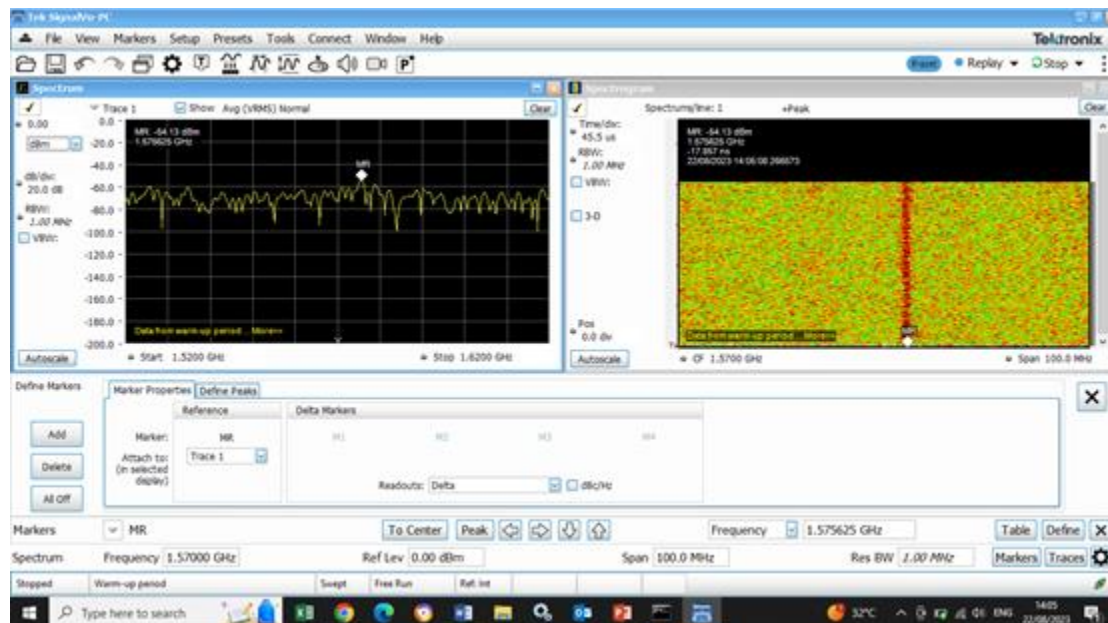


Figure 140: Spectrum After GPS Transmission

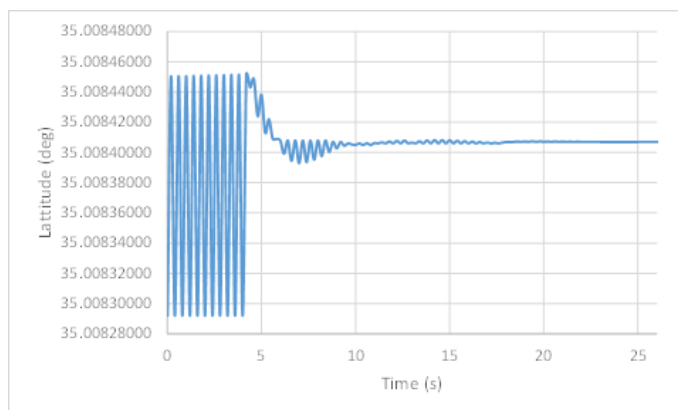


Figure 141: Latitude GPS Received by Autopilot

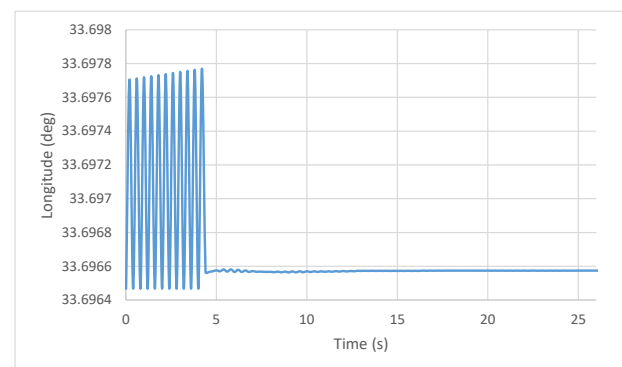


Figure 142: Longitude GPS Received by Autopilot

Figures 141 and 142 show the GPS measurements recorded from the autopilot. As it can be seen from the altitude and longitude measurements for the first 5 seconds there were oscillations until the GPS signal was locked and stabilised. Even though both signals are steady still when further analysed they have an error in the range of 4.2 meters which is not acceptable for the proposed application. The error can be clearly seen from Figure 143 when both latitude and longitude are graphed together.



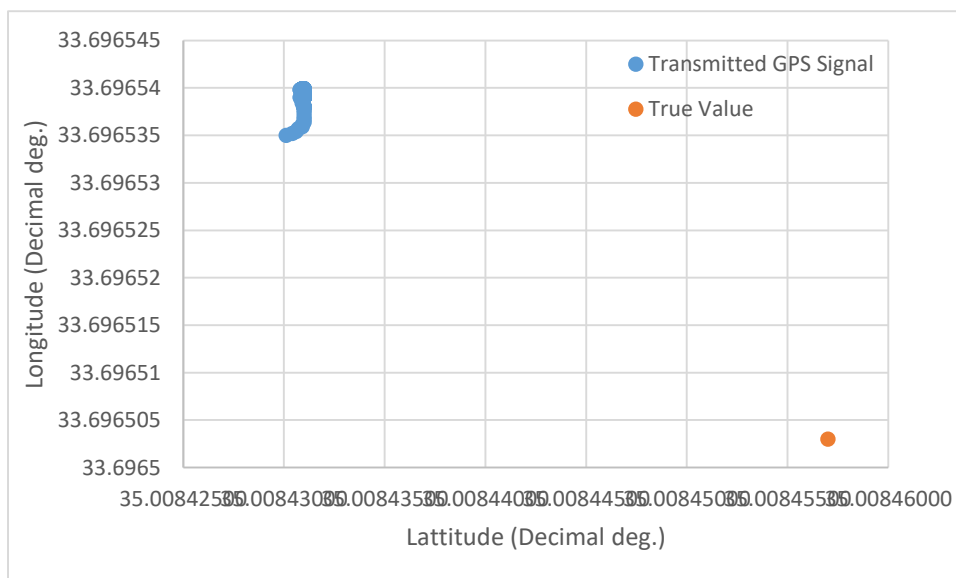


Figure 143: Longitude versus Latitude

On Figure 143, the transmitted GPS signal has high precision since they are all gathered together, On the other hand though the same signal is far away from the True Value. This error was calculated to 4.2 meter. Hence, it was decided to implement a kalman filter to improve the performance.

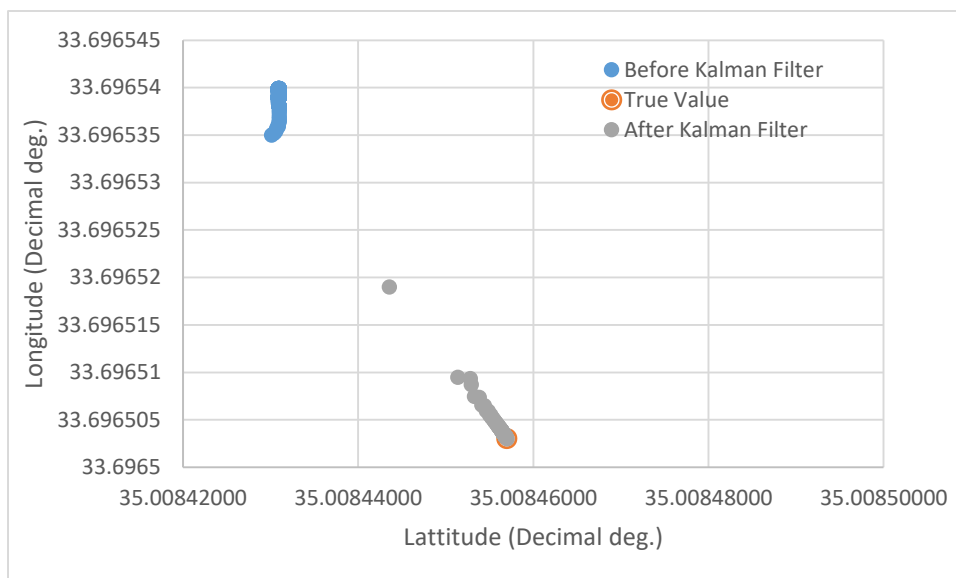


Figure 144: GPS Signal before and After Kalman Filtering

Figure 144 shows the results from when the kalman filtering was implemented. As can be seen using kalman filtering the values converge towards the true value. The new accuracy is examined on Figures 145 and 146.

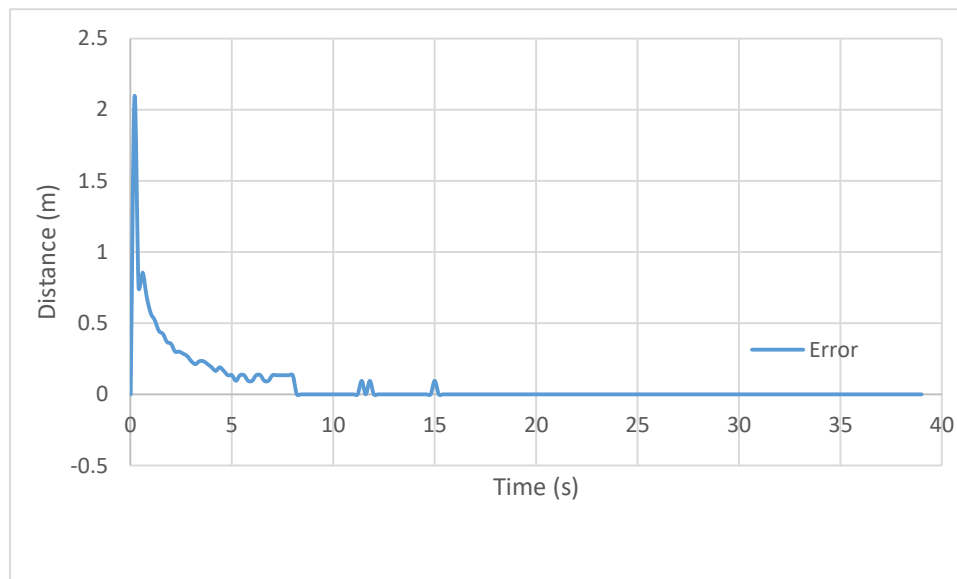


Figure 145: Error after Kalman Filtering

As shown from Figure 145 using kalman filtering the error significantly reduces to less than 0.5 meters. A closer look from figure 146 shows that in 3 seconds the error is reduced to 0.23 m and in 4.5 seconds it reduces to 0.15 m.

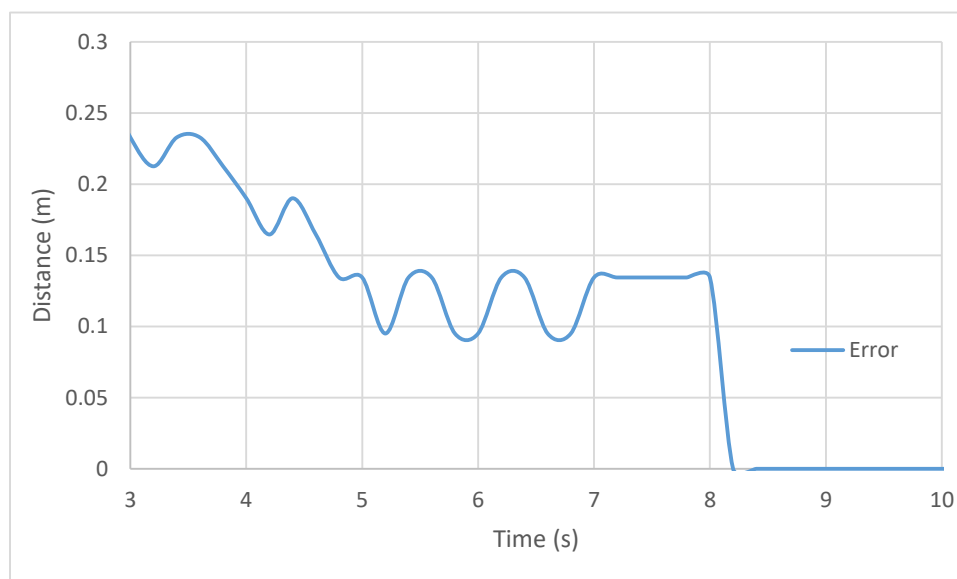
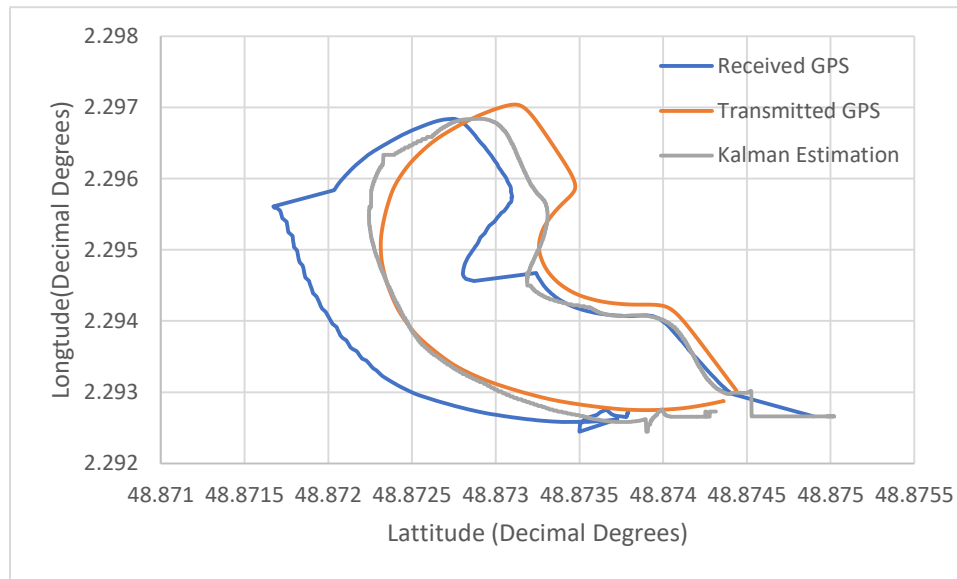


Figure 146: Closer Look at the Error after Kalman Filtering

Worth noting that ardupilot autopilot uses 5Hz GPS receiver. For newer autopilot which use GPS receiver that operate at 10 and 20 Hz then the kalman filter would converge in 1.5 and 0.75 seconds respectively. Figure 147 shows the results from dynamic GPS retransmission.



*Figure 147: Dynamic GPS Retransmission*

Figure 147 shows the results from dynamic GPS retransmission. Kalman filtering improves the accuracy. On the other hand when there was a turn the kalman filter needed time to converge. This is not a big issue because on a moving vehicle an IMU (inertial Measurement Unit) can be used to provide extra inputs to the kalman filter for faster convergence and accuracy.

This section proved the feasibility for Indoor Positioning by GPS Conversion and Retransmission. As shown on the analysis of the experimental results, the accuracy is in the range of 0.15 m which is in the sub-meter range required by the set KPI. **Both the feasibility and accuracy successfully meet the KPI requirements.**